REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* 30-06-2007	**2. REPORT TYPE** Final Technical Report	**3. DATES COVERED** *(From - To)* 1 Apr 2004 – 31 Mar 2007

4. TITLE AND SUBTITLE Cooperation and Consensus Seeking for Teams of Unmanned Air Vehicles	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER FA9550-04-1-0209
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) McLain, Tim (mclain@byu.edu) Beard, Randy (beard@byu.edu)	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Brigham Young University Provo, UT 84602	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research (AFOSR) 875 N. Arlington St., Rm. 3112 Arlington, VA 22203 Scott Wells/NL	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFOSR
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFRL-SR-AR-TR-07-0282

12. DISTRIBUTION AVAILABILITY STATEMENT

DISTRIBUTION A: Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report describes research in the area of cooperative control of unmanned air vehicles carried out at Brigham Young University with the support of AFOSR during the period from April 2004 to March 2007. Research efforts have centered on the development of cooperative control methods and consensus-seeking strategies for multi-vehicle teams with experimental validation of those methods.

15. SUBJECT TERMS

cooperative control, coordination, consensus, unmanned air vehicles, UAVs, unmanned aircraft systems, UAS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Lt Col Scott R. Wells
Unclassified	Unclassified	Unclassified	Unclassified	67	**19b. TELEPONE NUMBER** (***Include area code***) (703)

COOPERATION AND CONSENSUS SEEKING FOR TEAMS OF UNMANNED AIR VEHICLES

Final Report

AFOSR GRANT NO. FA9550-04-1-0209

Principal Investigators

Timothy W. McLain
Department of Mechanical Engineering
Brigham Young University
Provo, Utah 84604 USA
voice: (801) 422-6537
fax: (801) 422-0516
email: mclain@byu.edu

Randal W. Beard
Department of Electrical and Computer Engineering
Brigham Young University
Provo, Utah 84604 USA
voice: (801) 422-8392
fax: (801) 422-0201
email: beard@byu.edu

Abstract

This report describes research in the area of cooperative control of unmanned air vehicles carried out at Brigham Young University with the support of AFOSR during the period from April 2004 to March 2007. Research efforts have centered on the development of cooperative control methods and consensus-seeking strategies for multi-vehicle teams with experimental validation of those methods.

Objectives

Our research focused on cooperative control of unmanned air vehicles (UAVs). Under this AFOSR supported effort, we have sought to accomplish three primary objectives: 1) extend our current theoretical approach to encompass broader classes of cooperative control problems, 2) develop strategies for building consensus among a team of vehicles with inconsistent sensory information, and 3) experimentally demonstrate the effectiveness of our cooperative control strategies on a team of small UAVs.

The technical content of this report is divided into four main sections. Section 1 discusses an application of our coordination variable approach to a new cooperative surveillance problem. In the cooperative surveillance problem, a team of UAVs is used to monitor the periphery of a region of interest. The periphery can change in time requiring cooperation among the UAVs to utilize resources effectively. The region of interest could be a national border, a spreading forest fire, a growing toxic plume, or a group of fleeing enemy combatants.

Our recent work on the development of consensus seeking strategies for teams of UAVs is detailed in Sections 2 and 3. Specifically, we have developed a method for dealing with relative uncertainty between agents. We have shown that consensus algorithms can be developed that are stable in the presence of communication and sensor noise. In addition, we have extended well-known results on average consensus to account for more reasonable communication architectures.

We have conducted numerous multiple UAV experiments. We have demonstrated several cooperative timing scenarios using three UAVs as shown in Section 4. These experiments represent significant progress in two ways: First, they demonstrate the practical validity of our theoretical cooperative control developments. Second, they demonstrate a significant increase in the robustness and reliability of the multiple UAV systems in use. Flying multiple UAVs has directed us towards important research problems of practical interest. Solving some of these important problems has enabled increased reliability in multiple UAV operations.

Contents

1 Perimeter Surveillance

Perimeter surveillance algorithms form the basis for effective monitoring in a number of applications including monitoring oil spills [1], contaminant clouds [2], algae bloom [3], forest fires [4, 5], and border security [6, 7]. The literature in this area can roughly be decomposed into two main groups: sensor technology used for perimeter detection; and algorithms used to gather data along the perimeter effectively.

Sensors that have been investigated for small fixed-border scenarios, such as warehouse surveillance, include cameras [8], ultrasound [9], and radar [10]. In [8], the authors discuss algorithms that use image data from multiple cameras to determine a perimeter breach. Peralta [9] uses a chain of ultrasound sensors with a simple detection scheme to identify border crossings. Research has also been done using existing airport radar equipment to identify when people or vehicles come too close to runways [10]. For spill monitoring and other dynamic perimeter scenarios, surveillance vehicles are equipped with chemical concentration sensors [11], infrared cameras [5], or standard optical cameras [1].

We are particularly interested in borders that are of unknown shape and size and possibly changing in time such as would be encountered in a forest fire or chemical spill monitoring scenario. Additionally, we do not exclude large borders where communication range will limit the possible interaction of the team. We will assume that UAV agents have the proper sensor suite to detect changes in the perimeter and track the edge of the perimeter. We will not focus on the necessary sensor technology to do this, but rather on the algorithms that will allow a team of agents to monitor a perimeter in a decentralized fashion. Perimeter surveillance using multiple UAVs has the advantage of operating in a wide variety of circumstances such as changing perimeters (spill monitoring, forest fire surveillance) or very large perimeters (border patrol). In addition, small UAVs are relatively inexpensive and can be rapidly deployed.

A number of researchers have investigated similar problems of monitoring/tracking changing perimeters with autonomous vehicles. The MDARS project [12] is a joint effort between the Army and Navy that networks multiple ground robots to cooperatively monitor a fixed perimeter near critical storage facilities. A team of robots are equipped with coarse obstacle detection sensors and a high precision narrow field of view sensor to find and track objects that have breached the perimeter [7]. The entire team of vehicles communicates to a central location where sensor data is fused and waypoint commands are issued [6]. Our work differs from MDARS in that we do not require team agents to be in constant communication with a centralized controller; rather, agents are frequently outside of the communication range of the other team members and must monitor the perimeter in a decentralized manner.

Teams of unmanned water vehicles have been proposed as a way to track algae bloom and oil spills. Bertozzie et al. in [13] present an algorithm for monitoring a perimeter with multiple agents when each is equipped with a concentration sensor. When the sensor detects the presence of the chemical, the vehicle turns in one direction; in the absence of chemical detection, the vehicle turns in the opposite direction. In this way, an agent weaves around the perimeter of the spill while communicating the perimeter crossing points to form a complete picture of the perimeter. A simple spacing law adjusts the speed of the vehicles to allow the team to spread out along the perimeter. The algorithm has been shown to work in hardware testbed experiments with virtual perimeters [14].

Clark and Fierro propose a similar method for oil spill perimeter tracking using multiple vehicles [1]. A fleet of vehicles is deployed and will search the region and communicate to team members when the perimeter is located. Agents will approach the perimeter and begin to track it in a predetermined direction. Spacing of the vehicles is accomplished by adjusting linear velocity. Hardware experiments using a camera sensor on wheeled robots is shown to validate the algorithm. In both this approach and the one proposed by Bertozzi et al [13], neither the efficiency nor the convergence of the algorithms are shown analytically. In addition, neither explicitly address the problem of limited communication range.

Susca, Martinez and Bullo address the issue of approximating a changing border with a set of interpolation points [15]. As the team agents traverse the perimeter, they update the points that describe the perimeter to best fit a polygon to the shape of the perimeter. Their algorithm is shown to converge and relies only on communication between immediate neighbors.

In this section, we will present an algorithm for perimeter surveillance that: (1) is completely decentralized, (2) is provable convergent to the optimal behavior in finite-time, (3) explicitly accounts for communication range limitations, and (4) allows for changing perimeters. The primary advantages to a decentralized approach are scalability and inherent system robustness. Since agents only make decisions based on neighbor interactions, the required communication bandwidth and computation is fixed irrespective of the total number of agents on the team. Decentralization is inherently robust since each agent makes decisions with its available information without a need to receive directions from a central location. This eliminates single points of failure and allows a system to adapt naturally to changes in team size. Agents can be inserted and deleted from the team at any time and the system will adjust since each agent will maneuver to find its new neighbors. This allows agents to leave the team for high priority tasks, such as following a perimeter breach, or in case of accident or refueling.

In addition to being fully decentralized, our approach is optimal at steady-state and has finite-time convergence. Additionally, our approach requires very little communication bandwidth and accounts for UAV kinematic constraints. The algorithm is limited to constant velocity vehicles that travel along the border and due to its decentralized nature, any global information that may be available is not exploited. For missions where robustness is valued more than efficiency, our approach is a natural fit. Since it guarantees optimality in steady-state and finite-time convergence, only missions that have strict efficiency requirements would not be well-suited.

The perimeter surveillance problem is posed in Sections 1.1 and 1.2. Section 1.3 presents our solution using a coordination variable [16] approach and compares it to both averaging and centralized solutions. The method is extended to changing perimeters in Section 1.4 and to account for constrained UAV turning radius in Section 1.5. Simulation and hardware results are presented in Section 1.6. Finally, Section 1.8 gives our conclusions.

1.1 Problem Formulation

The objective of the cooperative perimeter surveillance problem is to cooperatively gather information about the state of the perimeter and transmit that data to a central base station with as little delay and at the highest rate possible. There are a number of factors that complicate the perimeter surveillance problem including:

1. Perimeter topology
2. Communication constraints
3. Team logistics
4. UAV capability.

Perimeter Topology. A perimeter may be static, such as a well-defined border, or changing in time, such as a chemical spill or forest fire. A perimeter can be composed of a web of segments and nodes that must be monitored, such as a set of city streets or a network of paths in the mountains, although we do require the graph representing the perimeter to be strongly connected, i.e. not disconnected. An area surveillance problem can sometimes be posed as a perimeter surveillance problem by constructing a path that covers the full area (e.g. zamboni pattern) and then monitoring that path as a perimeter. The perimeter location need not be known *a priori*, but when this is the case we assume that the UAVs have the sensor capacity to detect and follow the perimeter autonomously.

Communication Constraints. Small, inexpensive UAVs often have limited communication bandwidth and short communication range. In scenarios where the perimeter is very large or terrain causes line-of-sight problems agents may frequently be out of communication range of the base station and neighboring UAVs. Additionally, the gathered data may require significant time to transmit when a UAV is in communication range of its neighbors (e.g. complete video footage).

Team Logistics. UAVs have limited flight time and must be periodically refueled. In many cases, a UAV may be re-tasked to investigate a perimeter breach. Hardware failures and hazardous flying conditions may unexpectedly remove a UAV from involvement. A perimeter surveillance solution should be robust to failures and allow for interruptions such as reassignment and refueling.

UAV Capability. The maneuverability of the UAV agents also effects the monitoring of a perimeter. We assume that the UAVs are equipped with an autopilot similar to the one described in [17]. The autopilot maintains constant altitude and each UAV on the team is given a unique altitude assignment. The autopilot has been tuned so that the closed-loop system exhibits a first-order response to roll and airspeed commands. The equations of motion for a single UAV can be written as

$$\dot{p}_N = V\cos\psi + w_N \tag{1}$$
$$\dot{p}_E = V\sin\psi + w_E \tag{2}$$
$$\dot{\psi} = \frac{g}{V}\tan\phi \tag{3}$$
$$\dot{V} = \alpha_V(V^c - V) \tag{4}$$
$$\dot{\phi} = \alpha_\phi(\phi^c - \phi), \tag{5}$$

where $\mathbf{p} = (p_N, p_E)^T$ is the inertial position of the UAV, ψ, ϕ, and V are the heading, roll angle, and airspeed of the UAV, g is the gravitational constant, $\mathbf{w} = (w_N, w_E)^T$ is the windspeed, and V^c and ϕ^c are the airspeed and roll angle commands given to the autopilot. The first order response of the autopilot to airspeed and roll angle commands are quantified by the parameters α_V and α_ϕ. In addition to these kinematics, a constraint on roll angle $-\phi_{\max} \leq \phi \leq \phi_{\max}$ is enforced to ensure the safety of the UAV. The presence of wind and the roll angle constraint impair the maneuverability of the team agents.

Developing a perimeter surveillance algorithm that accounts for these complications and efficiently gathers data about the perimeter state is not trivial. We reduce the general problem to a more manageable, but still applicable, one and present the team behavior that efficiently solves that problem in Section 1.2. Section 1.3 then introduces and proves an algorithm for reaching the desired behavior while accounting for most of the complications.

1.2 Linear Perimeter Surveillance

We reduce the general perimeter surveillance problem of Section 1.1 to the linear surveillance problem by requiring that the perimeter to be monitored be represented as a single path between two points, i.e. the perimeter is homeomorphic to a line. This removes perimeters that are circular or are connected in a web-type format. However, an arbitrary perimeter can be posed as a linear perimeter by constructing a single tour that traverses all segments in the original perimeter. In practice, a surveillance mission will have a base of operations where information about the perimeter is analyzed and team agents are refueled and launched. Circular perimeters can be treated as linear perimeters with both endpoints at the base of operations.

A linear perimeter imposes a natural order to the team where each agent has at most two immediate neighbors along the perimeter. By requiring that neighbors physically meet to transmit information, any size of communication range is allowed. In practice, the sensor limitations of an agent will require it to monitor the entire perimeter up to its neighbor regardless of whether the agents can communicate earlier, therefore we assume that agents must meet to transmit data. Agent meetings can be extended to allow the transmission of large amounts of data. Loss or reassignment of team agents are quickly noticed by the change in the neighborhood of affected agents.

Team planning is done by considering agents as point agents that move at uniform constant velocity along the perimeter (see Figure 1). Corresponding UAV agents follow their reference points along the perimeter as described in Section 1.5. Point agents can reverse direction instantaneously and always do so when the end of the perimeter is encountered. Communication between agents is only allowed when the agents are "touching", i.e. when they occupy the same physical location. One way to visualize the problem is to imagine beads sliding along a string.

The performance of a particular monitoring algorithm can be measured by the latency associated with information about points along the perimeter. Let P be the length of the perimeter and let the perimeter

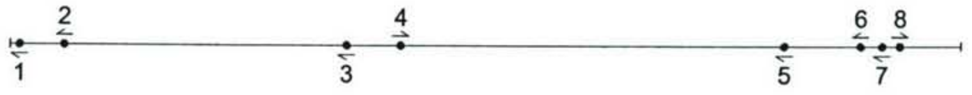
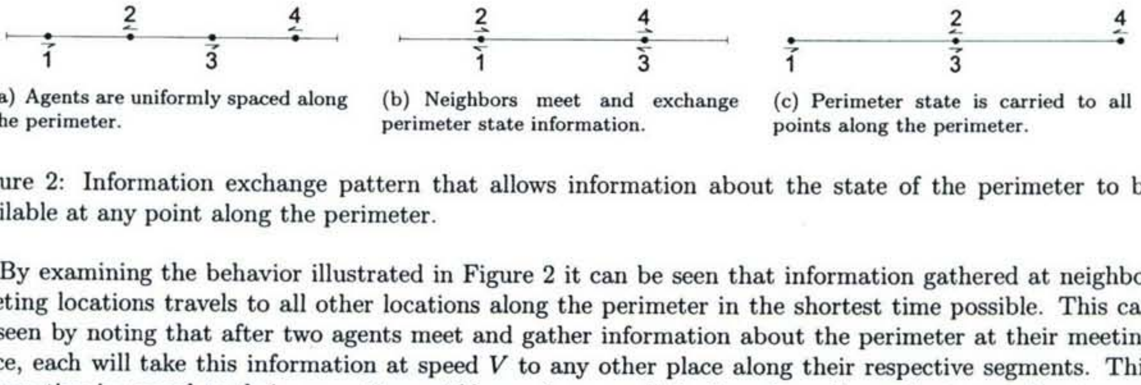


Figure 1: Example scenario where 8 agents monitor a linear perimeter.

be defined as a line along the x-axis beginning at $x = 0$ and continuing until $x = P$. Since we assume that the agents travel at uniform velocity, V, and data transmission only occurs when the agent is in immediate physical proximity, the soonest information about point x_0 is available to a recipient at the base of operations $(x = 0)$ is in x_0/V seconds. The minimum latency profile is obtained when an agent starts at the far end of the perimeter and travels to the base of operations at which time it transmits all the perimeter information.

Note that adding more agents cannot decrease the *latency* of the gathered information as seen at the base of operations since information can only travel as fast as a single agent. However, increasing the number of agents on the team increases the *refresh rate* of the perimeter state. Intuitively, spacing agents equally so that the refresh rate is constant will yield the most efficient method of perimeter monitoring. This configuration can be achieved by tasking each agent to travel to the end of the perimeter and then monitor the entire perimeter as it returns to the base while launching agents at $2P/N$ intervals where N is the number of agents on the team. As agents monitor the perimeter while traveling to the base of operations they pass agents traveling to the end of the perimeter to begin monitoring. These meetings occur at equally spaced intervals of size P/N. Rather than have agents traverse the entire perimeter equally spaced, each can be responsible for a segment of length P/N and pass its gathered information to its neighbors achieving the same overall latency profile and refresh rate.

Consider the behavior of a team of four agents as shown in Figure 2. The agents are uniformly distributed along the perimeter (Figure 2(a)) and each agent meets its neighbors at the end of its segments (Figures 2(b) and 2(c)). This oscillatory behavior of the agents requires that the team be synchronized not only in space (equally distributed), but also in time (meet neighbors at the end of segments).



(a) Agents are uniformly spaced along the perimeter.

(b) Neighbors meet and exchange perimeter state information.

(c) Perimeter state is carried to all points along the perimeter.

Figure 2: Information exchange pattern that allows information about the state of the perimeter to be available at any point along the perimeter.

By examining the behavior illustrated in Figure 2 it can be seen that information gathered at neighbor meeting locations travels to all other locations along the perimeter in the shortest time possible. This can be seen by noting that after two agents meet and gather information about the perimeter at their meeting place, each will take this information at speed V to any other place along their respective segments. This information is passed to their respective neighbors who carry it further along the perimeter, again at speed V. Therefore, the information gathered at the neighbor meeting locations is carried to all other points along the perimeter at the highest possible speed.

Definition 1.1 (Low-Latency Exchange Configuration). *Consider a team of N agents monitoring a linear perimeter of length P defined as a line along the x-axis from $x = 0$ to $x = P$. Order the agents from the left end of the perimeter $1 \dots N$. Consider two sets of team agent locations on the perimeter:*

1. *Agent $n \in 1 \dots N$ is located at $\lfloor n + \frac{1}{2}(-1)^n \rfloor P/N$*
2. *Agent $n \in 1 \dots N$ is located at $\lceil n + \frac{1}{2}(-1)^n \rceil P/N$*

The low-latency exchange configuration *is the behavior realized by the team when oscillating between these two team locations at speed V.*

As indicated earlier, the low-latency exchange configuration is the ideal behavior for a team of agents monitoring a linear perimeter and it will be the desired steady-state behavior of the decentralized algorithm presented in Section 1.3. In addition to converging to the low-latency exchange configuration, the algorithm will address deletion and insertion of team members and variable length perimeters.

1.3 Decentralized Solution

This section derives a decentralized algorithm to reach the low-latency exchange configuration defined in Definition 1.1. One way to approach such a problem is to determine the *coordination variables* [18] or minimum amount of information necessary to achieve cooperation. Three such critical pieces of information for this problem are: (1) the perimeter length, (2) the number of agents on the left side of the perimeter relative to a given agent, and (3) the number of agents on the right side of the perimeter relative to a given agent. When each agent has coordination variables that match reality, then each will be able to compute the perimeter segment for which it is responsible. The first step in the decentralized solution is to ensure that when each agent has the proper values, that coordination will be achieved.

Consider an algorithm where each agent assumes responsibility for a portion of the perimeter and escorts any of its intruding neighbors to their shared segment border. The following algorithm ensures that if each agent has coordination variable values that match reality (i.e. each agents knows the length of the perimeter, the total number of agents on the team, and its position in the team), then the low-latency exchange configuration will reached.

Algorithm 1: Neighbor Escort

```
if rendezvous with neighbor then
    Calculate shared border position.
    Travel with neighbor to shared border.
    Set direction to monitor own segment.
else if reached perimeter endpoint then
    Reverse direction.
else
    Continue in current direction.
```

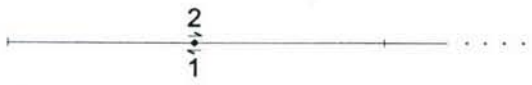
For every consecutive pair of agents, there is a single position where their segments border each other. When each agent has a knowledge of the length of the perimeter and its order in the team, then the endpoints of its responsible segment are easily computed. If an agent has determined that its position in the team is n and the length of the perimeter is P, then the segment for which that agent is responsible has endpoints at $(n-1)P/N$ and nP/N. The endpoint shared with a neighbor is the shared border position to which both will travel together in the first phase of Algorithm 1. In other words, each agent escorts its neighbors to the position at which they should have met had they been in perfect synchronization. Since agents only reverse direction at perimeter ends and when they finish escorting neighbor agents, each agent is guaranteed to meet its neighbors.

Theorem 1.1. *Let the perimeter length and number of agents be fixed. If all agents have coordination values that match reality, then Algorithm 1 ensures that the low-latency exchange configuration is achieved after time $2T$ has passed where T corresponds to the time required for one agent to travel the length of the perimeter.*

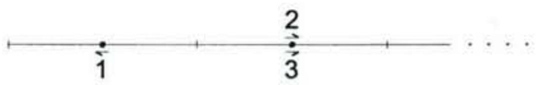
Proof: Team agents can initially be positioned anywhere along the perimeter and be traveling either in the positive or negative direction (recall that constant uniform velocity is assumed). Since each agent has correct coordination variables, then each can calculate the segment along the perimeter for which it is responsible.

Agents are guaranteed to meet both neighbors since Algorithm 1 only commands agents to reverse direction at a perimeter endpoint or when concluding a neighbor escort.

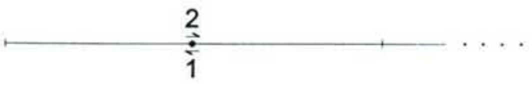
For N agents monitoring a border of length P, order segments of size P/N from the left edge of the perimeter $1, \dots, N$ and let the number of each agent correspond to the segment for which it is responsible. Consider the effect of Algorithm 1 on the leftmost agent (agent 1). Once agent 1 has escorted its right neighbor to their shared border, then no agent to the right of agent 1 will ever travel along segment 1 again. This can be seen by noting that after agents 1 and 2 split at their shared boundary *both* will travel the length of one segment to get to the opposite end of their respective segments. If agent 2 meets agent 3 along the way, then agents 2 and 3 will continue to their shared border before agent 2 reverses direction, as in Figure 1.3. Therefore, agent 2 will travel at least one segment length away from the boundary between segments 1 and 2. Since both travel at a uniform constant velocity, then agent 1 will arrive back at the border between segments 1 and 2 at the same time or before agent 2, but never after. Now consider agent 2 after it has been escorted by agent 1 to their shared boundary. Since by this time agent 2 never ventures into segment 1, the border between agents 1 and 2 can be regarded as a fixed perimeter endpoint for agent 2. The same analysis now holds if we consider agent 2 the leftmost agent in a set of $N-1$ agents. Observe that the same argument holds starting with the rightmost agent and considering all agents to the left. Therefore, there is a time τ after which all agents are only found on their respective segments. This implies that the low-latency exchange configuration of Section 1.2 has been reached.



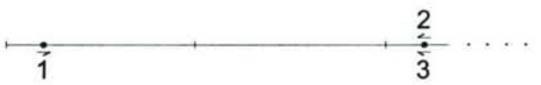
(a) Agents 1 and 2 separate at their shared border.



(b) Agent 2 encounters agent 3 earlier than expected and escorts it to their shared border before reversing direction.



(c) Agents 1 and 2 separate at their shared border.



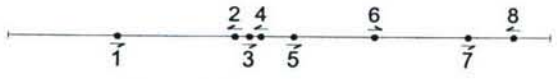
(d) Agent 2 encounters agent 3 later than expected and escorts it to their shared border before continuing on to meet agent 1.

The worst case situation occurs when all agents are stacked infinitesimally close at one end of the perimeter and are traveling toward the other. Once T has passed all agents are at the opposite end of the perimeter where they meet both neighbors. Each pair will travel to their shared borders which for the farthest pair will require a travel time less than T. Therefore, the steady-state behavior will be achieved before time $2T$. ■

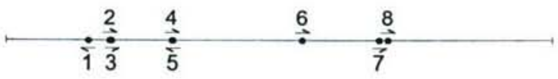
Figure 3 shows two simple scenarios with 8 agents spreading out over a fixed perimeter where each agent begins with correct coordination variables. The positions of agents along the perimeter is indicated vertically with the time axis shown horizontally. The lattice structure indicates that the desired steady-state behavior has been reached since agents turn around at precisely their desired neighbor rendezvous locations. Note that the agents require very few meetings with other agents to converge to the proper configuration.

1.3.1 Comparison with Centralized Algorithm

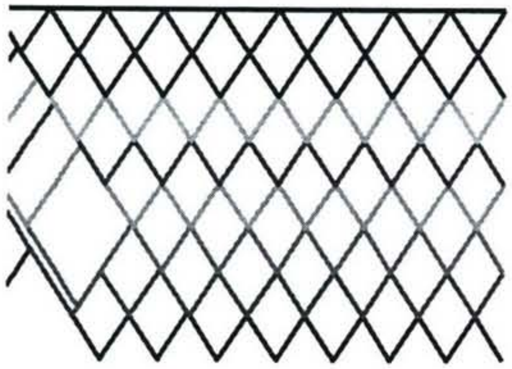
To understand the value of Algorithm 1, it is useful to compare its performance with other methods of perimeter surveillance. A centralized method for reaching the low-latency exchange configuration is to compare the initial positions of the team with all possible team locations in the low-latency exchange configuration and find the one that requires the shortest convergence time.



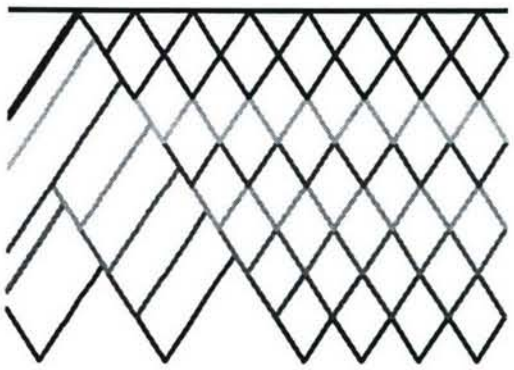
(a) Initial positions and directions for a group of 8 agents in scenario A.



(b) Initial positions and directions for a group of 8 agents in scenario B.



(c) Position of agents along the perimeter over time for scenario A.



(d) Position of agents along the perimeter over time for scenario B.

Figure 3: Team behavior in two scenarios for point agents whose behavior is governed by Algorithm 1. The position of agents along the perimeter is indicated vertically with the time axis shown horizontally. The lattice structure indicates that the desired steady-state behavior has been reached.

Let $\mathcal{S}$ be the set of team positions during the desired steady-state operation where an element $s \in \mathcal{S}$ consists of N positions, s_i corresponding to the position of agent i in the low-latency exchange configuration. Note that the set of all team configurations that satisfy the low-latency exchange configuration can be parameterized by the position of the first agent

$$s_i = \frac{i-1}{N} + \begin{cases} \left(\frac{1}{N} - s_1\right) & \text{if } i \text{ is even} \\ s_1 & \text{otherwise .} \end{cases} \tag{6}$$

Therefore, if the position of the first agent is known in the low latency configuration, then for a perimeter of unit length, s_1 is on the interval $[0, \frac{1}{N}]$ and all other positions can be calculated using Equation (6). The centralized method is to command the team located at p_i, $i = 1 \ldots N$ to converge to s^* where

$$s^* = \arg\min_{s \in \mathcal{S}} \max_{i=1 \ldots N} |p_i - s_i| \quad . \tag{7}$$

In other words, the optimal solution is to pick the closest team position in the low-latency configuration to the current position of the agent. During the transition from the initial position to the nearest low-latency configuration position s^*, agents reach their correct position and wait there until the remaining team members have reached their respective positions.

In the worst case scenario with all agents located at one end of the perimeter, the centralized algorithm converges in T, twice as fast as the decentralized method. Figure 4 shows a comparison of the centralized algorithm and Algorithm 1. Note that the centralized algorithm requires agents to wait or loiter at the proper location until all agents have reached s^*. This is indicated by the straight lines in Figure 4.

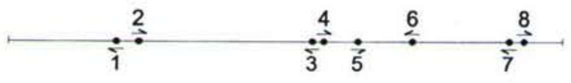
Monte-Carlo simulations indicate that the centralized algorithm reaches the low-latency exchange configuration on average $0.67T$ faster than the decentralized method (standard deviation of $0.17T$). The maximum time difference between the centralized algorithm and Algorithm 1 was $0.998T$ with theoretical worst case difference of T. The centralized algorithm requires complete knowledge of the state of the team and explicit cooperation of all team members. The value of Algorithm 1 is that its performance is comparable to the optimal solution in speed, but is implemented in a decentralized, robust way.

1.3.2 Comparison with Consensus Method

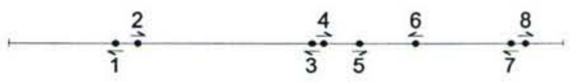
The second method to which we compare Algorithm 1 is a distributed consensus algorithm modified for perimeter surveillance. The standard consensus problem is when a group of agents, each with a variable of arbitrary initial value, reaches agreement on the value of that variable through repeated interaction with other agents on the team. For example, one method of coming into consensus is for each agent to repeatedly average its associated variable with those communicated from its immediate neighbors. If the interaction graph among the team contains a spanning tree, then the variable of each agent will asymptotically approach a constant shared value and the team is said to asymptotically reach consensus [19].

Adapting a consensus method to the perimeter surveillance problem involves defining the value associated for each agent and a strategy for updating those values. Let the length of the segment for which an agent is responsible be the value associated with that agent. Consider the rendezvous of two agents on the perimeter. When agents meet, they communicate the length of their respective segments and average to find the midpoint of their shared segment. Both travel together to the midpoint of their shared segment [20] and split apart with an updated value for how much of the perimeter each is responsible for. For every pair of agents, their shared segment is defined by endpoints determined by the locations each agent met its previous neighbor.

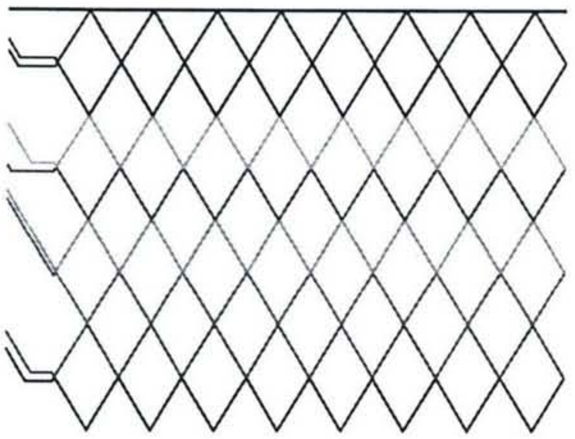
The difficulty with this method lies in the realization that the value to which the team will converge *must* be P/N where P is the length of the perimeter; otherwise, agents would be continuously overlapping or neglecting part of the perimeter. A specialization of the general consensus problem to the average consensus problem can be made which ensures that the team will converge to the exact average of the initial values. The only remaining difficulty is initializing the system so that the segment lengths associated with the team of agents sum to P. We do this by assuming that agents are launched with a value of zero with the exception of the first agent who travels to the end of the perimeter and initializes its value to P. This had a number of



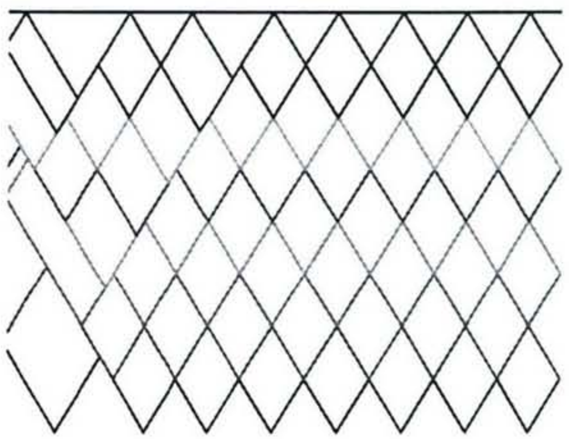
(a) Initial positions and directions for a group of 8 agents.



(b) Initial positions and directions for a group of 8 agents.



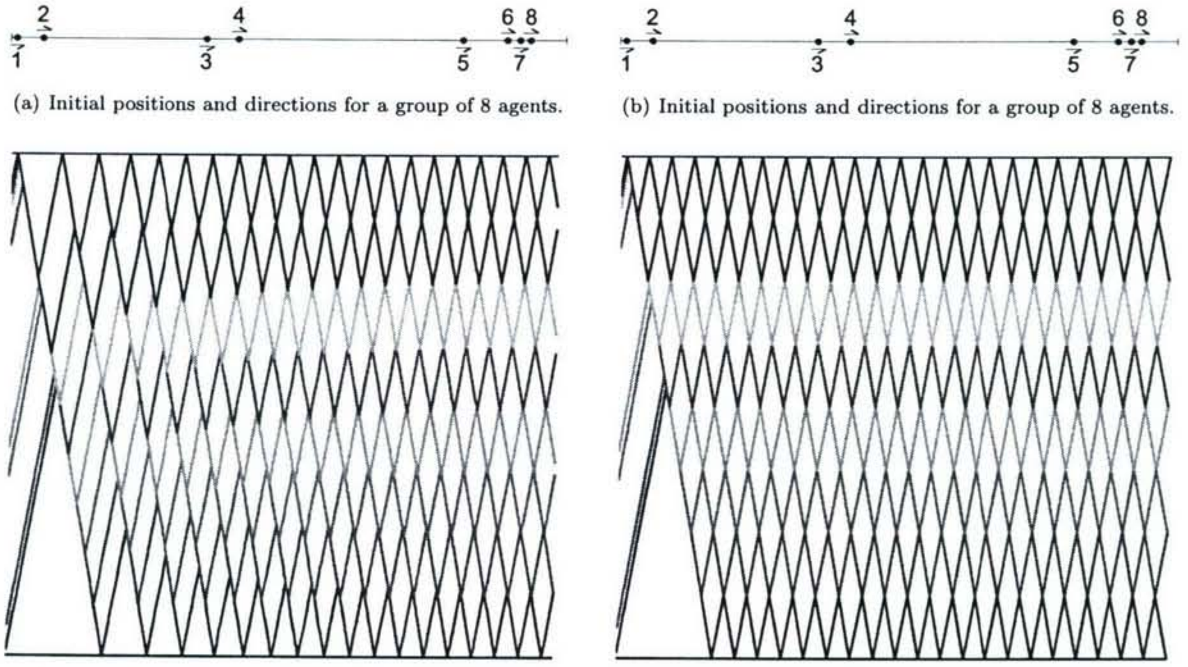
(c) Position of agents along the perimeter over time when using the centralized algorithm (7).



(d) Position of agents along the perimeter over time when using Algorithm 1.

Figure 4: Team behavior in a comparison of Algorithm 1 and the centralized algorithm (7). The position of agents along the perimeter is indicated vertically with the time axis shown horizontally. The lattice structure indicates that the desired steady-state behavior has been reached. Straight lines indicate that an agent is maintaining its current position along the perimeter. Note that the centralized algorithm requires agents to wait for the rest of the team to settle into the optimal starting configuration while Algorithm 1 reaches the low-latency exchange configuration after some interaction time.

consequences: although the algorithm can account for arbitrary perimeter length, the perimeter must remain fixed; secondly, loss of an agent during the mission will remove its segment length from the knowledge of the team. In each case, the prerequisites for average consensus would be violated and the team would fail to converge. Additionally, convergence is, in general, asymptotic in nature rather than in finite-time as Algorithm 1 guarantees. Figure 5 shows the performance of the average consensus algorithm compared to Algorithm 1. In addition to the above limitations of fixed perimeter length and asymptotic convergence, the consensus method seems to exhibit poor transient response.



(a) Initial positions and directions for a group of 8 agents.

(b) Initial positions and directions for a group of 8 agents.

(c) Position of agents along the perimeter over time when using the centralized algorithm (7).

(d) Position of agents along the perimeter over time when using Algorithm 1.

Figure 5: Team behavior in a comparison of Algorithm 1 and a consensus method (7). The position of agents along the perimeter is indicated vertically with the time axis shown horizontally. The lattice structure indicates that the desired steady-state behavior has been reached. Note that the consensus method converges asymptotically while Algorithm 1 reaches the low-latency exchange configuration in finite time.

Algorithm 1 relies only on interactions of an agent with its immediate neighbors on the perimeter yet converges to the low-latency exchange configuration in finite time. In Section 1.4 we show that decentralized nature of the algorithm allows the team to accommodate loss or reassignment of agents. In the event of a perimeter breach, an agent can be assigned to follow the intruder while the rest of the team reconfigures to monitor the border in its absence. Since the algorithm converges in finite-time, the loss in perimeter coverage is quickly compensated for. This same natural reconfiguration behavior is desirable in the event of refueling and agent loss due to hazardous conditions.

1.4 Changing Perimeters

Theorem 1.1 ensures finite time convergence to the low-latency exchange configuration of Section 1.2 when the coordination variables match reality. By adding the ability to update the coordination variables, Algorithm 1 can be modified to ensure that there will be a time when the coordination variables are correct. This will allow the team to naturally compensate for agent reassignment or loss and perimeter growth.

Each agent maintains local variables that track the amount of perimeter distance and number of agents on each side. These coordination variables are updated when meeting with another agent on the team – both agents have knowledge about the number of agents and perimeter length on their respective sides. If the perimeter and number of agents is fixed, then the coordination variables will eventually be consistent among the team since agents are guaranteed to meet both neighbors. Once the coordination variables are correct, Theorem 1.1 ensures that the desired steady-state behavior will be achieved. Note that the same method used to update the coordination variables can also be used to detect changes in the perimeter or insertion/deletion of team members. Algorithm 2 includes the local coordination variables and the logic required to update each agent. In Algorithm 2 the variable 'side' is used to indicate the positioning of an agent relative to its neighbor, i.e. 'left' or 'right' along the perimeter. The term '$\overline{\text{side}}$' is simply the opposite of 'side'.

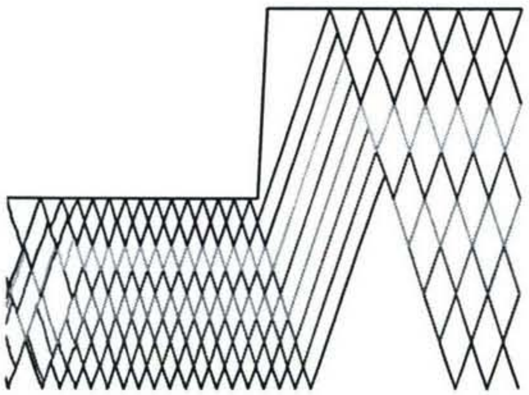
Algorithm 2: Variable Neighbor Escort

```
 Variables:
     $P_{\text{left}}$ length of perimeter to left of agent
     $P_{\text{right}}$ length of perimeter to right of agent
     $N_{\text{left}}$ number of agents to left of agent
     $N_{\text{right}}$ number of agents to right of agent
 if rendezvous with neighbor then
     Communicate $P_{\text{side}}$ and $N_{\text{side}}$.
     Update $P_{\overline{\text{side}}}$ and $N_{\overline{\text{side}}}$.
     Calculate shared border position.
     Travel with neighbor to shared border.
     Set direction to monitor own segment.
 else if reached perimeter endpoint then
     Reverse direction.
     $P_{\text{side}} = 0$
     $N_{\text{side}} = 0$
 else
     Continue in current direction while updating $P_{\text{side}}$.
```

Algorithm 2 operates in the same manner as Algorithm 1 only with the additional steps of communicating and updating the coordination variables. For example, consider two agents starting from opposite ends of the perimeter, each without knowledge of the other. Let agent 1 start at $x = 0$ and agent 2 start at $x = P$, but let the launch time of agent 2 be delayed with respect the launch of agent 1. As each agent progresses along the perimeter, it keeps track of the distance traveled from launch. When the two agents finally meet, agent 1 updates N_{right} to be equal to one plus the number of agents to the right of agent 2 and P_{right} equal to P_{right} communicated from agent 2; similarly, agent 2 updates N_{left} and P_{left} from the communication from agent 1. At this point, the coordination variables match reality and Theorem 1.1 ensures that the low-latency exchange configuration will be reached in finite time.

The applicability of Algorithm 2 lies in the fact that each agent has finite memory. Since the coordination variables are updated with the most recent information gathered, past information does not affect team behavior. This finite memory property allows the team to adapt to changes in perimeter and team size. Figure 6 shows agent tracking on a perimeter with a step change in size and a perimeter with sinusoidal growth. The algorithm accommodates step changes in perimeter size, but also allows good tracking for other types of perimeter growth. Note that agents do not have any knowledge *a priori* of the perimeter length or number of agents on the team. The coordination variables are updated through repeated interactions with other team members.

Algorithm 1 can also be extended to account for long communication events. In a perimeter imaging scenario, agents survey the perimeter segment for which they are responsible and when each meets its neighbor it must transmit large amounts of data. In this case, agent meetings cannot be instantaneous, rather


(a) Positions of agents over time for step change in perimeter length.

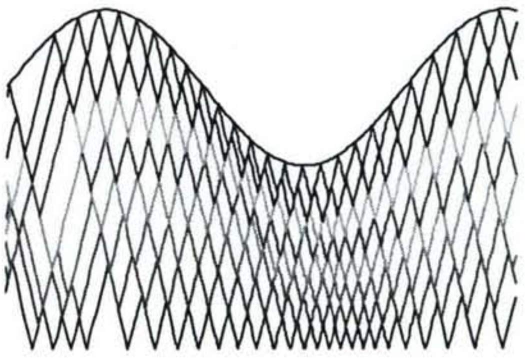

(b) Positions of agents over time for sinusoidal perimeter growth.

Figure 6: Team behavior of agents tracking changing perimeters using Algorithm 2 to continuously update the coordination variables. Agents learn the size of the perimeter and number on the team through repeated interaction with other team members.

a fixed amount of time is allotted for agents to loiter at the meeting location to allow longer communication events. After an agent finishes escorting its neighbor, both loiter together for a pre-determined amount of time. Figure 7 shows a scenario involving long communication events.

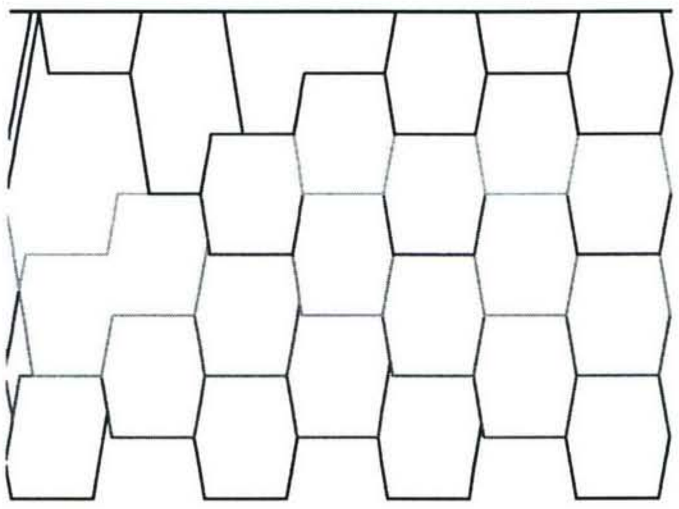


Figure 7: Example scenario where Algorithm 1 is modified to account for long rendezvous timing.

1.5 UAV Agents

Algorithm 1 developed the motion of the reference points for a team of UAVs to follow to achieve the low-latency exchange configuration. In practice, the reference point generalization is only followed when agents are involved in a rendezvous with another agent. Between meetings, the center of the constant airspeed UAV is considered the point along the perimeter. However, since a UAV has a constrained turning radius, it cannot precisely follow a reference point that can instantaneously turn around. The purpose of this section is to investigate the limitations of Algorithm 1 when applied to perimeter surveillance using a team of UAVs.

In Section 1.2 agents are modeled as points that could communicate only when touching. Now consider UAVs flying at constant velocity with nominal turning radius R. To complete a U-turn with such a turning radius requires that the UAV be aware of the need to turn around a distance of $\frac{7}{6}\pi R$ before the actual rendezvous [21]. Since both UAVs must have a buffer of this distance in order to complete a turn-around, the communication radius of each UAV must be greater than $\frac{7}{3}\pi R$. When a UAV becomes aware of an immanent rendezvous, it can calculate where it should begin its U-turn maneuver to maintain the point-agent motion of Algorithm 1.

The constrained turning radius also affects how many agents can monitor a perimeter without being interrupted during a turn. The smallest segment is one on which a UAV can complete two U-turns consecutively, so a perimeter of length P being monitored by UAVs with radius R can have at most $\lfloor \frac{3}{7\pi R} P \rfloor$ agents.

Consider a scenario with N UAV agents that have a large enough communication radius to perform the U-turn maneuver as described above. Also, let the perimeter be large enough so that in steady-state the segment for which a UAV is responsible is large compared to the turning radius of the vehicle (specifically, larger than $\frac{7}{3}\pi R$). Under what conditions can we ensure that during the transient period, no UAV is interrupted while performing a turning maneuver? In other words, when is the distance between consecutive direction reversals smaller than the distance required to perform those maneuvers?

It is easy to show that when the scenario allows for uninterrupted steady-state behavior and the agents have consistent coordination variables, that no UAV is interrupted during the transient period. However, during the time when the team is forming a consistent set of coordination variables, the distance between consecutive direction changes can be made arbitrarily short with proper choice of initial positions and directions of the agents. A sufficient condition to ensure that no UAV is interrupted in the transient period is to enforce a separation distance of at least $\frac{7}{3}\pi R$ at launch. In the general case, Algorithm 1 could be further modified to allow interrupted turns to translate into effective perimeter growth. These effects will die out as soon as the coordination variables reach consistency.

1.6 Simulation Results

To verify the feasibility of implementing Algorithm 1 on a team of UAVs, a high fidelity simulation is performed. Each UAV is simulated with full 6 degree of freedom dynamics model with aerodynamic parameters that match the small UAVs flown at BYU [17]. The simulation scenario involved three UAVs monitoring a changing perimeter composed of 4 waypoints with a total length of 1.46 km. Each UAV is equipped with autopilot software that enables accurate waypoint tracking [17] with a turning radius of approximately 50 meters. The communication model allows UAVs to communicate only to adjacent neighbors who are inside the communication range of approximately 370 meters, the minimum distance necessary.

The simulation scenario starts with only two of the three UAVs being launched. Each agent starts without any knowledge of the number of agents on the team nor the perimeter length. Even though the perimeter is defined by predetermined waypoints, we require the UAVs to initially treat the perimeter length as unknown. After about 400 seconds, a step change in the perimeter length occurred by adding an additional waypoint, followed by another change a short time later. At approximately 900 seconds in simulation time, the third UAV was launched. Before the simulation terminated, the team experienced two more changes in the perimeter length, one at each end.

Figure 8 shows the simulation results by plotting the normalized position of each UAV along the length of the perimeter. Note that in the regions where the team should already be locked into the ideal configuration, some position overlap is still observed. This is caused by the inability of the UAVs to perform the U-turn maneuver precisely, and resulted in a disturbance to the system. However, the overall behavior of the team was as expected, with the agents reaching the desired steady-state behavior quickly and reacting

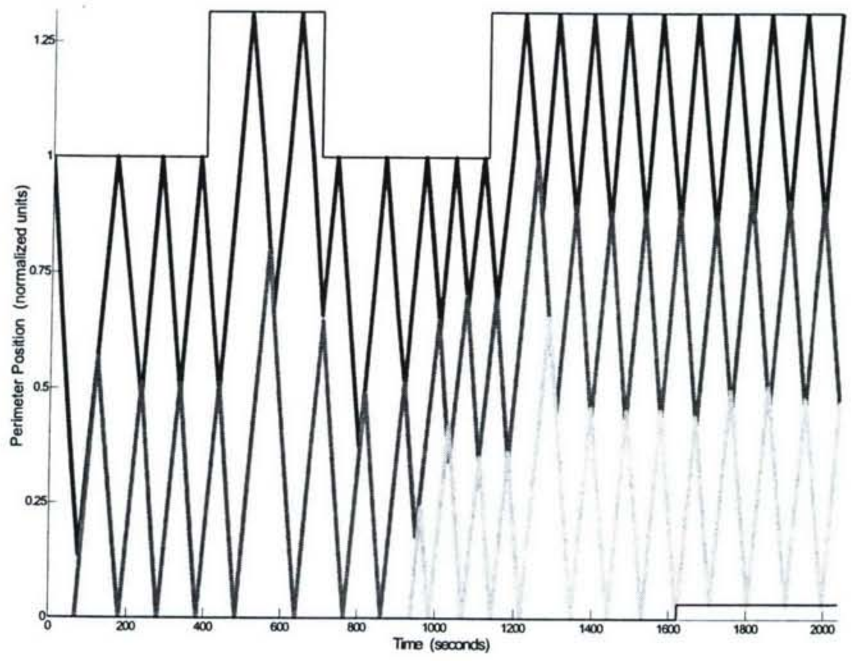


Figure 8: Simulation results showing the normalized position of each UAS along the perimeter. Changes to the perimeter length occurred at approximately 400, 700, 1100, and 1600 seconds. The third agent was introduced at approximately 950 seconds. The sharp peaks are a result of the coordination variables being reset.

appropriately to step changes in both the perimeter length and team size.

It should be noted that even though the UAVs cannot turn around instantaneously, the position plot in Figure 8 shows the agents making sudden changes in direction. This is a result of the UAV resetting the coordination variables when it begins to execute the turn around command.

1.7 Flight Test Results

The decentralized cooperative-surveillance algorithm was further validated by hardware flight tests using the experimental testbed described [17]. Figure 9 displays the normalized position of two UAVs along the perimeter while Figure 10 shows the inertial position plots that were generated from the actual telemetry files of the UAVs. The latter figure demonstrates the algorithm by showing (a) some initial condition for the two agents, (b) the first rendezvous, (c) the turn-around at the shared border, (d) the first meeting of the perimeter endpoints, (e) the second rendezvous, and (f) the second meeting of the perimeter endpoints.

The algorithm was initiated at approximately 50 seconds, after the two agents had passed each other. The first UAV (blue) turned around immediately while the second agent (red) traveled to the shared border before turning around. At this point the agents had reached the steady-state configuration. Similar to the simulation results, there was some overlap in position between the two agents. This is a result of the inability of the UAVs to complete a precise U-turn maneuver. It should also be noted that the shared-border position of the two agents appears to be around 60% of the perimeter length instead of the theoretically predicted 50%. This deviation was caused by wind pushing the second agent, thereby enabling Agent 2 to cover more distance than Agent 1. Wind speeds during the flight tests were estimated at 35% of the airspeed of the UAVs. Despite the disturbance of the wind, the agents were still able to effectively distribute themselves evenly along the perimeter.

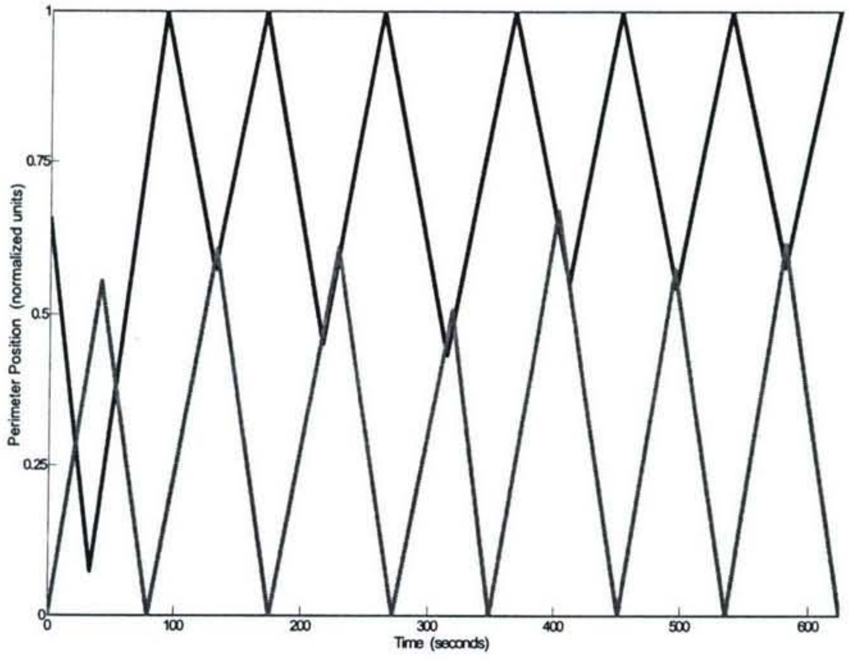
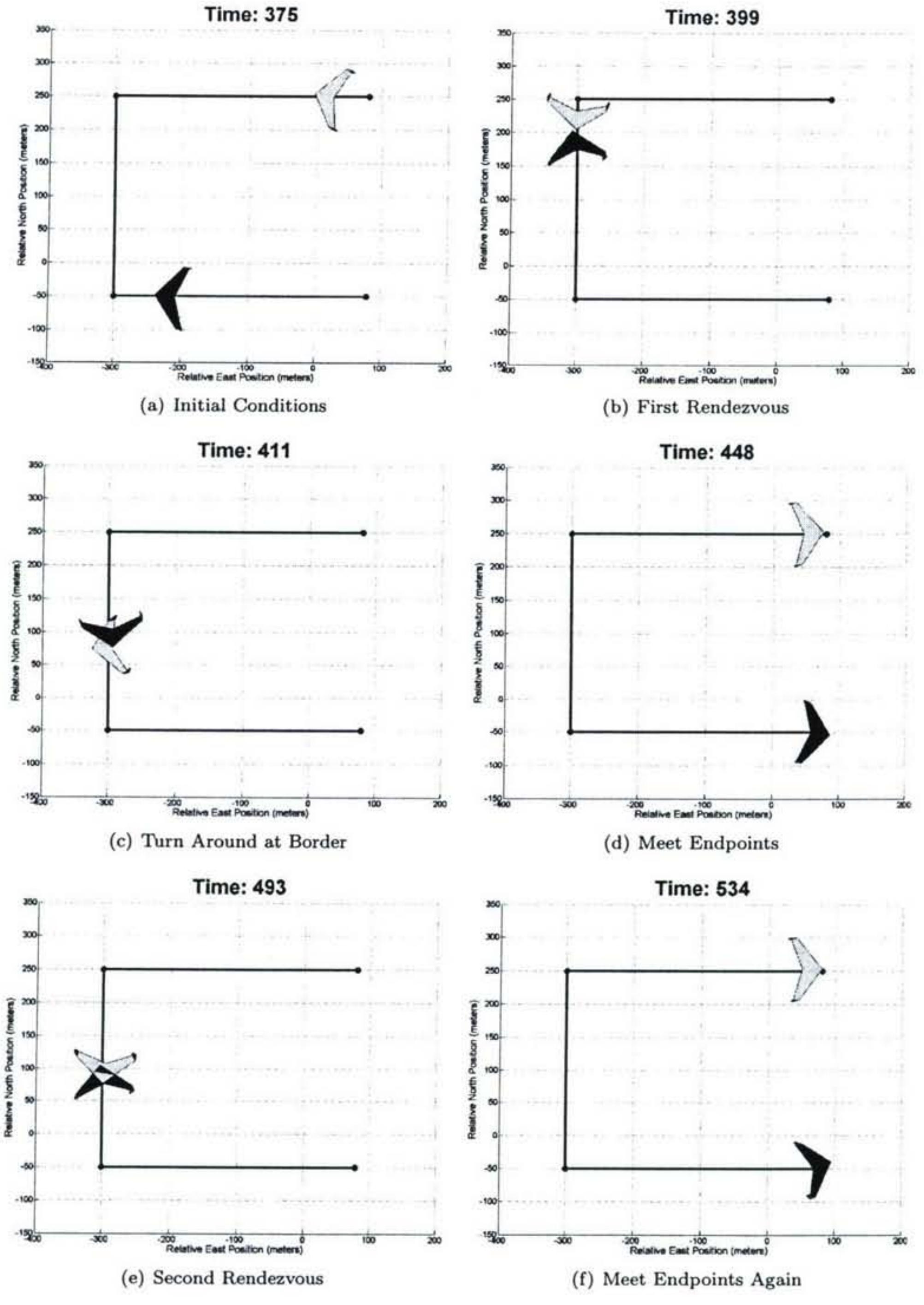


Figure 9: Experimental results showing the normalized position of each UAS along the perimeter. The decentralized cooperative-surveillance algorithm was started at approximately 50 seconds.

1.8 Conclusions

This section has presented a decentralized algorithm for perimeter surveillance that converges in finite time. By sharing information regarding the perimeter length and number of team members, each agent obtains a consistent set of coordination variables that allows the decentralized algorithm to operate. Advantages include the ability to monitor changing perimeters, account for dynamic insertion and deletion of team members, and operate with small communication range in a decentralized manner. Simulation and flight tests were performed to validate the effectiveness of the algorithm.



(a) Initial Conditions

(b) First Rendezvous

(c) Turn Around at Border

(d) Meet Endpoints

(e) Second Rendezvous

(f) Meet Endpoints Again

Figure 10: Various plots generated from the actual telemetry files of the UASs collected during the experimental flight tests. These demonstrated the functionality of the distributed spread algorithm, where (a) are the initial conditions, (b) is the first rendezvous, (c) is the turn-around at the shared border, (d) is the first meeting of the perimeter endpoints, (e) is the second rendezvous, and (f) is the second meeting of the perimeter endpoints.

2 Kalman Consensus Strategies and Their Application to Cooperative Control

During the last two decades there has been a dramatic paradigm shift in the way that computer systems are designed: moving from centralized mainframe computers to networks of less capable, but much less expensive, personal computers. In much the same way, replacing large, expensive, monolithic vehicles with teams of networked vehicles, promises less expensive, more capable systems. In addition, there are applications where a team of vehicles can accomplish objectives that would be impossible for a single vehicle. For example, a formation of networked spacecraft could be used to synthesize a space-based interferometer with base-lines reaching tens to hundreds of kilometers [22, 23]. With teams of vehicles, much of the design complexity is shifted from mechanical hardware design to software that regulates the interaction of the team.

In recent years, there has been significant interest and research activity in the area of coordinated and cooperative control [24, 25, 26, 27, 28, 29, 30, 31]. Much of this work assumes the availability of global team knowledge, and/or the ability to plan group actions in a centralized manner.

Centralized coordination techniques are suitable if each member of the team has the ability to communicate to a centralized location or if the team is able to share information via a static fully connected network. On the other hand, real-world communication topologies are usually not fully connected. In many cases they depend on the relative position of the vehicles and on other environmental factors and are therefore dynamically changing in time. In addition, wireless communication channels are subject to multi-path, fading and drop-out. Therefore, cooperative control in the presence of real-world communication constraints, becomes a significant challenge.

In a recent article we argued that "shared information is a necessary condition for cooperation" [16]. Shared information may take the form of common objectives, common control algorithms, relative position information, or a world map. If this assertion is true, then information exchange becomes a central issue in cooperative control. In this article, we will refer to the information that is necessary for coordination as the *coordination information* or *coordination variable* [32]. In the presence of an unreliable, dynamically changing communication topology, it is not possible for all of the vehicles to have access to identical coordination information. Suppose that a particular cooperation strategy has been devised and shown to work if the team has global access to the coordination information. Cooperation will occur if each member on the team has access to the same information.

As an example, consider the meet-for-dinner problem introduced in [16]. In this problem, a group of friends decide to meet for dinner at a particular restaurant but fail to specify a precise time to meet. On the afternoon of the dinner appointment, each individual realizes that they are uncertain about the time that the group will meet for dinner. A centralized solution to this problem is for the group to have a conference call, to poll each individual regarding their preferred time for dinner, and to average the answers to arrive at a time that the group will meet for dinner. However, this centralized solution requires that a conference line is available, and that the time of the conference call is known to the group. Since, whatever algorithm was used to convey the time of the conference call to the group, could also have been used to convey the time to meet for dinner, the central problem remains.

The information variable in this example is the time that the group will meet for dinner. The particular time is not what is important, but rather that each individual in the group has a consistent understanding of that information. A decentralized solution to the problem would be for each individual to call, one at a time, a subset of the group. Given his current estimate of the meeting time, the individual might update his estimate of the meeting time to be a weighted average of his current meeting time and that of the person with whom he is conversing. The question (which will be answered in this section) is under what conditions this strategy will enable the entire team to converge to a consistent meeting time.

Therefore, if a centralized solution to a cooperation problem, with its associated coordination information, has been devised, then two additional questions must be addressed. First, what algorithms should be employed to ensure that the team is converging to a consistent view of the coordination information in the presence of an unreliable, dynamically changing communication topology? Second, if the action of the group is based on the (dynamically changing) coordination variable, will the cooperative control algorithm be robust with respect to the transient error in the coordination variable across the team?

Convergence to a consistent view of the coordination variable in the presence of an unreliable, dynamically changing communication topology is called the consensus problem. Consensus problems have recently been addressed in [33, 34, 35, 36, 37, 16, 38, 39], to name a few. In [34], sufficient conditions are given for consensus of the heading angles of a group of agents under undirected switching interaction topologies. In [35], average consensus problems are solved for a network of integrators using directed graphs. In [16] and [38], an algebraic graph approach is used to show necessary and/or sufficient conditions for consensus of information under time-invariant and switching interaction topologies respectively. In [36], a set-valued Lyapunov function approach is used to consider discrete-time consensus problems with unidirectional time-dependent communication links.

Previous consensus seeking results reported in the literature do not explicitly account for agent confidence in their instantiation of the coordination variable. Most results assume that each individual in the group has identical confidence in their instantiation of the coordination variable. However, there are many cases where some individuals on the team will have access to better information than others. In cases like these, the consensus algorithm needs to be biased to favor agents with better information. For example, if a team of UAVs is tasked with tracking the location of a group of ground vehicles, the quality of information will be proportional to the relative sensing distance. UAVs that have recently flown close to a ground vehicle should be considered more reliable than those that are sensing from a greater distance, or whose information is old. As another example, in the meet-for-dinner problem described above, if one individual is considered more reliable than the others, his/her information should be weighted more heavily when making the team decision.

The primary contribution of this section is to derive continuous-time and discrete-time consensus strategies, based on a Kalman-filter structure, that asymptotically achieves consensus in the presence of an unreliable, dynamically changing communication topology, giving proper weight to individuals with greater certainty in their coordination variable. In addition, we will show that the Kalman consensus scheme is input-to-state stable (ISS) where the input is the communication noise on each channel and the state is the consensus error between each pair of agents. The ISS property will be exploited to develop a distributed multi-vehicle cooperative control solution to the cooperative timing problem.

UAV cooperative timing problems have been investigated recently in the context of battlefield scenarios where the UAVs are required to converge on the boundary of a radar detection area to maximize the element of surprise [40, 32, 24, 41, 42]. Cooperative timing problems also arise in refueling scenarios, fire and hazardous material monitoring, moving area of regard problems, and continuous surveillance problems. In this section we will investigate a simplified cooperative timing problem that must be accomplished in the presence of an unreliable, dynamically changing communication topology.

The section is organized as follows. In Section 2.1, we give an intuitive, non-rigorous derivation of the Kalman-like consensus strategies, and show their application to the meet-for-dinner problem. Section 2.2 contains the main technical results that shows that the proposed consensus strategies are convergent under certain mild conditions. In Section 2.3 we show that the Kalman consensus scheme is input-to-state stable. As a corollary, we show that most of the other consensus schemes proposed in the literature, are also ISS. In Section 2.4 the ISS property is exploited to develop a distributed solution to the cooperative timing problem. Finally, Section 2.5 contains our conclusions.

2.1 Kalman-filter Approach to Multi-Agent Consensus

The Kalman filter is used extensively to estimate a system's current state from imprecise measurement data [43, 44, 45]. It is well-known that the Kalman filter is an optimal estimator in the case of Gaussian statistics and that it is the best linear estimator in the case of other statistics [46]. Motivated by the Kalman filter scheme, we treat the final consensus value as the system state, which is unknown *a priori* but is the final equilibrium state that each agent in the group is expected to achieve. In the consensus problem, each agent has an estimate of the final consensus value. Communication from other agents regarding their estimate of the final consensus value will be regarded as measurement data. In this sense, each agent in the group performs its own estimate of the final consensus value based on the information available to it. Our goal is to guarantee that the information state of each agent achieves the final consensus value. In other words, the objective is to minimize the mean squared error between each agent's estimate of the coordination variable and the final consensus value. The error covariance matrix is interpreted as the confidence that each agent has in its current estimate of the coordination variable, where large covariance indicates low confidence, and small covariance indicates a high degree of confidence. In Section 2.1.1 we will derive the Kalman-consensus scheme for a continuous-time update scheme, and in Section 2.1.2 we will address the discrete-time case. Analytical properties of the algorithms will be derived in Section 2.2.

2.1.1 Continuous-time Consensus

System model and measurement model:
$\dot{x} = Ax + Bu + Gw$
$z = Hx + v$
$x(0) \sim (\bar{x}_0, P_0), w \sim (0, Q), v \sim (0, R)$
Assumptions:
$\{w(t)\}$ and $\{v(t)\}$ are white noise processes uncorrelated with $x(0)$ and with each other. $R > 0$.
Initialization:
$P(0) = P_0, \hat{x}(0) = \bar{x}_0$
Error covariance update:
$\dot{P} = AP + PA^T + GQG^T - PH^TR^{-1}HP$
Kalman gain:
$K = PH^TR^{-1}$
Estimate update:
$\dot{\hat{x}} = A\hat{x} + Bu + K(z - H\hat{x})$

Table 1: Continuous-time Kalman filter [47].

The standard continuous-time Kalman filter is summarized in Table 2.1.1 [47]. The objective of this section is to show how the Kalman filter equations can be used to derive a decentralized information consensus scheme.

Let $\xi^* \in \mathbb{R}^m$ be the *a priori* unknown information state over which the team is to form consensus. In other words, each information state ξ_i will converge to the consensus value ξ^* as $t \to \infty$. Note that the consensus value will depend not only on interaction topologies but on the weighting factors in the update schemes. In this section we will assume that the consensus state is a constant, which implies that the system dynamics are given by

$$\dot{\xi}^* = w,$$

where, with reference to Table 2.1.1, $A = 0$, $B = 0$, $G = I_m$, and $E\{ww^T\} = Q$. In the following, we assume that $Q(t) > 0$ is uniformly lower and upper bounded.

Treat the ith information state ξ_i as the ith agent's estimate of ξ^* and suppose that the jth agent communicates ξ_j to the ith agent with transmission, or communication noise ν_{ij}. Also, let $g_{ij}(t)$ be a time-varying boolean variable that indicates the presence of an open communication channel from agent j to agent i at time t, i.e., $g_{ij}(t) = 1$ if information is communicated from j to i at time t and zero otherwise. Note that $g_{ii}(t) \triangleq 1$. Using these definitions, it is clear that the measurement model of the ith agent can be given by

$$z_i = \begin{pmatrix} g_{i1}(\xi_1 + \nu_{i1}) \\ \vdots \\ g_{iN}(\xi_N + \nu_{iN}) \end{pmatrix} = \begin{pmatrix} g_{i1}I \\ \vdots \\ g_{iN}I \end{pmatrix} \xi^* + \begin{pmatrix} g_{i1}(\xi_1 - \xi^* + \nu_{i1}) \\ \vdots \\ g_{iN}(\xi_N - \xi^* + \nu_{iN}) \end{pmatrix},$$

where, with reference to Table 2.1.1,

$$H_i^T = (g_{i1}I \quad \dots \quad g_{iN}I)$$

and

$$v_i = \begin{pmatrix} g_{i1}(\xi_1 - \xi^* + \nu_{i1}) \\ \vdots \\ g_{iN}(\xi_N - \xi^* + \nu_{iN}). \end{pmatrix}$$

If we define $P_i \triangleq E\{(\xi_i - \xi^*)(\xi_i - \xi^*)^T\}$ and assume that $E\{(\xi_i - \xi^*)(\xi_j - \xi^*)^T\} = 0$, where $i \neq j$, then

$$R_i \triangleq E\{v_i v_i^T\} = \begin{pmatrix} g_{i1}(P_1 + \Omega_{i1}) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & g_{iN}(P_N + \Omega_{iN}) \end{pmatrix},$$

where $\Omega_{ij} \triangleq E\{\nu_{ij}\nu_{ij}^T\}$ is assumed to be upper bounded.

Therefore, the error covariance update in Table 2.1.1 becomes

$$\dot{P}_i = -P_i H_i^T R_i^{-1} H_i P_i + Q = -P_i[\sum_{j=1}^{N} g_{ij}(P_j + \Omega_{ij})^{-1}]P_i + Q.$$

Similarly, the Kalman gain is given by

$$K_i = P_i H_i^T R_i^{-1} = (g_{i1}P_i(P_1 + \Omega_{i1})^{-1} \quad \cdots \quad g_{in}P_i(P_N + \Omega_{iN})^{-1}),$$

and the estimate update is given by

$$\dot{\xi}_i = K_i(z_i - H_i\xi_i) = K_i\left[\begin{pmatrix} g_{i1}(\xi_1 + \nu_{i1}) \\ \vdots \\ g_{in}(\xi_N + \nu_{iN}) \end{pmatrix} - \begin{pmatrix} g_{i1}I_m \\ \vdots \\ g_{iN}I_m \end{pmatrix}\xi_i\right] = \sum_{j=1}^{N} K_{ij}g_{ij}(\xi_j - \xi_i + \nu_{ij}),$$

where $K_i = [K_{i1}, K_{i2}, \cdots, K_{in}]$.

Summarizing, we have the following Kalman consensus scheme for the i^{th} agent:

$$\dot{P}_i = -P_i \left[\sum_j g_{ij}(t)(P_j + \Omega_{ij})^{-1} \right] P_i + Q \tag{8}$$

$$K_{ij} = P_i(P_j + \Omega_{ij})^{-1} \tag{9}$$

$$\dot{\xi}_i = \sum_{j=1}^{n} g_{ij}(t) K_{ij} \left((\xi_j + \nu_{ij}) - \xi_i \right). \tag{10}$$

Note that Eq. (8) indicates that the certainty of information increases with communication but decreases with the size of the process noise. In addition, the rate of increase in certainty for the i^{th} agent is inversely proportional to the certainty of the j^{th} agent and the communication noise. Note also that the Kalman gain K_{ij} is reduced if either the communication noise is large, or if the certainty of the j^{th} agent is small (hence P_j large). Note that Eq. (10) is similar to the continuous-time consensus schemes proposed in [34, 35, 16] except that the consensus gain K_{ij} is time-varying in (10), and the communication noise is explicitly included.

2.1.2 Discrete-time Consensus

System model and measurement model:
$x[k+1] = A[k]x[k] + B[k]u[k] + G[k]w[k]$
$z[k] = H[k]x[k] + v[k]$
$x(0) \sim (\bar{x}_0, P_{x_0}), w \sim (0, Q[k]), v[k] \sim (0, R[k])$
Assumptions:
$\{w[k]\}$ and $\{v[k]\}$ are white noise processes uncorrelated with x_0
and with each other. $R[k] > 0$.
Initialization:
$P[0] = P_{x_0}, \hat{x}_0 = \bar{x}_0$
Time update: (effect of system dynamics)
error covariance: $P[k+1]^- = A[k]P[k]A[k]^T + G[k]Q[k]G[k]^T$
estimte: $\hat{x}[k+1]^- = A[k]\hat{x}[k] + B[k]u[k]$
Measurement update: (effect of measurement $z[k]$)
error covariance: $P[k+1] = [(P[k+1]^-)^{-1} + H[k+1]^T R[k+1]^{-1} H[k+1]]^{-1}$
estimate: $\hat{x}[k+1] = \hat{x}[k+1]^- + P[k+1]H[k+1]^T R[k+1]^{-1}(z[k+1] - H[k+1]\hat{x}[k+1]^-)$

Table 2: Discrete-time Kalman filter [47].

The standard discrete-time Kalman filter is summarized in Table 2.1.2 [47]. Again assuming that ξ^* is constant we get

$$\xi^*[k+1] = \xi^*[k] + w[k],$$

where, with reference to Table 2.1.1, $A[k] = I$, $B[k] = 0$, $G[k] = I_m$, and $E\{w[k]w[k]^T\} = Q[k]$.

Again letting $\nu_{ij}[k]$ represent the communication noise, the measurement model for the i^{th} agent can be given by

$$
\begin{aligned}
z_i[k] &= \begin{pmatrix} g_{i1}[k]\left(\xi_1[k] + \nu_{i1}[k]\right) \\ \vdots \\ g_{iN}[k]\left(\xi_N[k] + \nu_{iN}[k]\right) \end{pmatrix} \\
&= \begin{pmatrix} g_{i1}[k]I \\ \vdots \\ g_{iN}[k]I \end{pmatrix} \xi^*[k] + \begin{pmatrix} g_{i1}[k]\left(\xi_1[k] - \xi^*[k] + \nu_{i1}[k]\right) \\ \vdots \\ g_{iN}[k]\left(\xi_N[k] - \xi^*[k] + \nu_{iN}[k]\right) \end{pmatrix},
\end{aligned}
$$

where, with reference to Table 2.1.1,

$$
H_i^T[k] = \begin{pmatrix} g_{i1}[k]I & \dots & g_{iN}[k]I \end{pmatrix}
$$

and

$$
v_i = \begin{pmatrix} g_{i1}[k]\left(\xi_1[k] - \xi^*[k] + \nu_{i1}[k]\right) \\ \vdots \\ g_{iN}[k]\left(\xi_N[k] - \xi^*[k] + \nu_{iN}[k]\right) \end{pmatrix}.
$$

If we define $P_i[k] \triangleq E\{(\xi_i[k] - \xi^*[k])(\xi_i[k] - \xi^*[k])^T\}$ and assume that $E\{(\xi_i[k] - \xi^*[k])(\xi_j[k] - \xi^*[k])^T\} = 0$, where $i \neq j$, then

$$
\begin{aligned}
R_i[k] &\triangleq E\{v_i[k]v_i[k]^T\} \\
&= \begin{pmatrix} g_{i1}[k](P_1[k] + \Omega_{i1}[k]) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & g_{iN}[k](P_N[k] + \Omega_{iN}[k]) \end{pmatrix},
\end{aligned}
$$

where $\Omega_{ij}[k] \triangleq E\{\nu_{ij}[k]\nu_{ij}[k]^T\}$.

Therefore, the time update in Table 2.1.2 becomes

$$
\begin{aligned}
P_i^-[k+1] &= P_i[k] + Q[k] \\
\xi_i^-[k+1] &= \xi_i[k].
\end{aligned}
$$

The measurement update is given by

$$
\begin{aligned}
P_i[k+1] &= \left[(P_i^-[k+1])^{-1} + H_i^T[k+1]R_i^{-1}[k][k+1]H_i[k+1]\right]^{-1} \\
&= [(P_i[k] + Q[k])^{-1} + \sum_{j=1}^{n} g_{ij}[k](P_j[k] + \Omega_{ij}[k])^{-1}]^{-1}, \\
\xi_i[k+1] &= \xi^-[k+1] + P_i[k+1]H_i^T[k+1]R_i^{-1}[k+1]\left(z_i[k+1] - H_i[k+1]\xi^-[k+1]\right) \\
&= \xi_i[k] + P_i[k+1]\left(\sum_{j=1}^{n} [g_{ij}[k](P_j[k] + \Omega_{ij})^{-1}[k]\left((\xi_j[k] + \nu_{ij}[k+1]) - \xi_i[k]\right)]\right).
\end{aligned}
$$

Summarizing, we have the following discrete-time Kalman consensus scheme for the i^{th} agent:

$$
P_i[k+1] = [(P_i[k] + Q[k])^{-1} + \sum_{j=1}^{n} g_{ij}[k](P_j[k] + \Omega_{ij}[k])^{-1}]^{-1}, \tag{11}
$$

$$
\xi_i[k+1] = \xi_i[k] + P_i[k+1]\left(\sum_{j=1}^{n} [g_{ij}[k](P_j[k] + \Omega_{ij})^{-1}[k]\left(\xi_j[k] + \nu_{ij}[k+1]) - \xi_i[k]\right)]\right). \tag{12}
$$

2.1.3 Meet for Dinner Example

To illustrate, consider the meet-for-dinner problem discussed in the introduction. Suppose that there are $N = 10$ agents who communicate with exactly one other individual, chosen randomly from the group, for a random length of time. After the communication has expired, the process is repeated. Figure 11 shows the

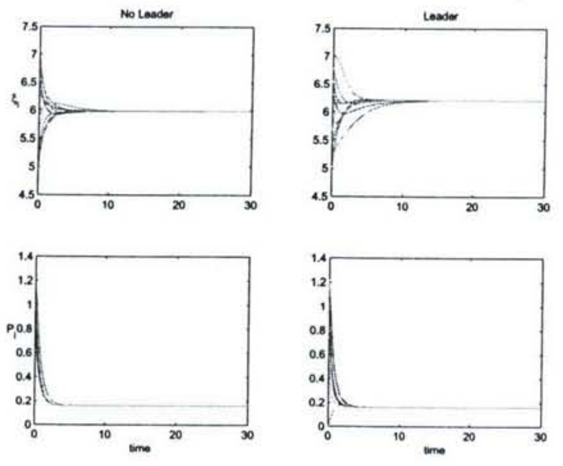


Figure 11: Continuous-time meet-for-dinner simulation. The subplot in the upper left shows the evolution of the coordination variable assuming that all agents begin with equally confident covariance. The subplot in the lower left shows the associated covariance. The subplots on the right show identical data where the agent with initial time $\xi_i = 7$ has an initial covariance of $P_i = 0.001$.

state and variance plots under the continuous Kalman consensus scheme (8)–(10) where the initial state is uniformly assigned. The subplots on the left show the arrival times and variance when the initial variances are uniformly assigned. The subplots on the right show the arrival times and variances when the variance of the agent with initial arrival time $\xi_i = 7$ is given an initial variance of $P_i = 0.001$, which is significantly lower than the other agents. Note that in this case, the final consensus value is influenced to a greater degree by this agent. Figure 12 shows similar plots using the discrete Kalman consensus scheme (11)–(12). Both the continuous-time and discrete-time simulations use the values $\Omega_{ij} = 0.1$, $Q = 0.1$.

2.2 Convergence Results

The objective of this section is to state some technical properties of the algorithms given in Eqs. (8)–(10) and Eqs. (11)–(12). For notational simplicity, we will focus on the case where each information state ξ^* is a scalar. The vector case reduces to the scalar case if P_{i0} is a diagonal matrix. The general case where P_{i0} is non-diagonal is currently a topic of research. In Section 2.2.1, we will introduce some notation and results from graph theory and non-negative matrices that will be used in the convergence arguments. In Section 2.2.2 we analyze the continuous-time case and in Section 2.2.3 we analyze the discrete-time case.

2.2.1 Preliminaries

Let $\mathcal{A} = \{A_i | i \in \mathcal{I}\}$, where $\mathcal{I} = \{1, 2, \cdots, n\}$, be a set of n agents among whom consensus is desired. A directed graph $\mathcal{G}$ will be used to model the interaction topology among these agents. In $\mathcal{G}$, the i^{th} vertex represents the i^{th} agent A_i and a directed edge from A_i to A_j denoted as (A_i, A_j) represents a unidirectional information exchange from A_i to A_j, that is, agent j receives information from agent i, $(i, j) \in \mathcal{I}$. If the information flows from agent i to agent j, agent i is called the parent of j, and agent j is called the child of i. A directed path in graph $\mathcal{G}$ is a sequence of edges $(A_{i_1}, A_{i_2}), (A_{i_2}, A_{i_3}), (A_{i_3}, A_{i_4}), \cdots$ in that graph. Graph

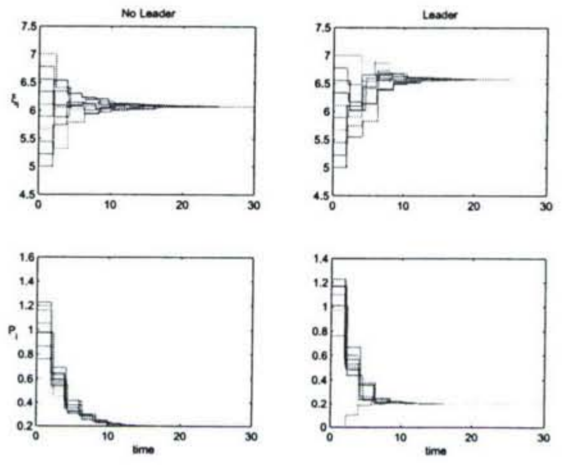


Figure 12: Discrete-time meet-for-dinner simulation. The subplot in the upper left shows the evolution of the coordination variable assuming that all agents begin with equally confident covariance. The subplot in the lower left shows the associated covariance. The subplots on the right show identical data where the agent with initial time $\xi_i = 7$ has an initial covariance of $P_i = 0.001$.

$\mathcal{G}$ is called strongly connected if there is a directed path from A_i to A_j and A_j to A_i between any pair of distinct vertices A_i and A_j, $\forall(i,j) \in \mathcal{I}$. A directed tree is a directed graph, where every node, except the root, has exactly one parent. A spanning tree of a directed graph is a directed tree formed by graph edges that connect all the vertices of the graph [48]. We say that a directed graph has a spanning tree if there exists a spanning tree that is a subset of the directed graph. Fig. 13 shows a directed graph with more than one possible spanning trees. The double arrows denote one possible spanning tree with A_5 as the parent. Spanning trees with A_1 and A_4 as the parent, are also possible. As a comparison, Figs. 14 shows two cases where the graph does not have a spanning tree.

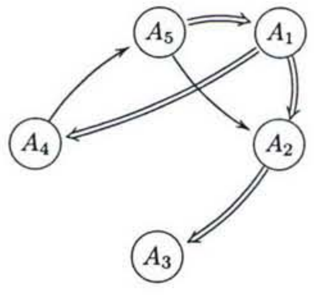


Figure 13: A directed graph that has more than one possible spanning trees, but is not strongly connected. One possible spanning tree is denoted with double arrows.

The interaction topology may change dynamically. Let $\bar{\mathcal{G}} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_M\}$ denote the set of all possible directed interaction graphs defined for $\mathcal{A}$. It is obvious that $\bar{\mathcal{G}}$ has a finite number of elements and that $\mathcal{G}(t) \in \bar{\mathcal{G}}$. The union of a set of directed graphs $\{\mathcal{G}_{i_1}, \mathcal{G}_{i_2}, \cdots, \mathcal{G}_{i_m}\} \subset \bar{\mathcal{G}}$ is a directed graph with vertices given by A_i, $i \in \mathcal{I}$ and edge set given by the union of the edge sets of $\mathcal{G}_{i_j}$, $j = 1, \cdots, m$. We will assume throughout the section that the interaction topology does not switch infinitely fast.

Let $M_n(\mathbb{R})$ represent the set of all $n \times n$ real matrices. Given a matrix $A = [a_{ij}] \in M_n(\mathbb{R})$, the directed graph of A, denoted by $\Gamma(A)$, is the directed graph on n vertices V_i, $i \in \mathcal{I}$, such that there is a directed edge

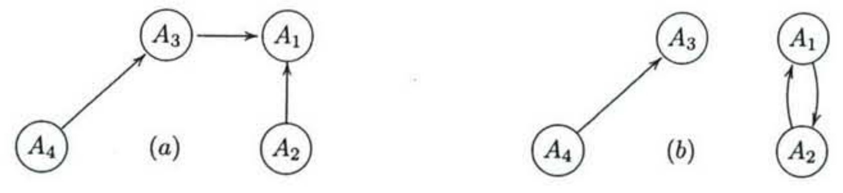


Figure 14: (a) A directed graph that has two leaders, and hence does not contain a spanning tree. (b) A directed graph that has two isolated groups, and hence does not contain a spanning tree.

in $\Gamma(A)$ from V_j to V_i if and only if $a_{ij} \neq 0$ [49]. For example, the directed graph of the matrix

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & a_{15} \\ a_{21} & 0 & 0 & 0 & a_{25} \\ 0 & a_{32} & 0 & 0 & 0 \\ a_{41} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{54} & 0 \end{bmatrix},$$

where $a_{ij} \neq 0$, corresponds to the graph in Fig. 13.

A matrix $A = [a_{ij}] \in M_n(\mathbb{R})$ is nonnegative, denoted as $A \geq 0$, if all its entries are nonnegative. Furthermore, if all its row sums are $+1$, A is said to be a (row) stochastic matrix [49]. A stochastic matrix P is called indecomposable and aperiodic (SIA) if $\lim_{n\to\infty} P^n = \mathbf{1}y^T$, where y is a column vector, and $\mathbf{1}$ denotes an $n \times 1$ column vector with all the entries equal to 1 [50]. For nonnegative matrices, $A \geq B$ implies that $A - B$ is a nonnegative matrix. It is easy to verify that if $A \geq \rho B$, for some $\rho > 0$, then the directed graph of B is a subset of the directed graph of A.

Two $n \times n$ nonnegative matrices are said to be of the same type if their zero elements are in the same locations [50]. We will use the notation $P \sim Q$ to denote that P and Q are of the same type.

Lemma 2.1. *Given $n \times n$ nonnegative matrices P, Q, R, and S, if $P \sim R$ and $Q \sim S$, then $(P+Q) \sim (R+S)$ and $PQ \sim RS$. Moreover, if a time-varying nonnegative matrix $M(t)$ with continuous entries is of a fixed type for $t \in [t_1, t_2]$, where $t_1 < t_2$, then $M(t) \sim \int_{t_1}^{t_2} M(t)dt$.*

Proof: Trivial. ∎

Let $\xi_i \in \mathbb{R}$, $i \in \mathcal{I}$, represent the i^{th} information state associated with the i^{th} agent. The set of agents $\mathcal{A}$ is said to achieve consensus asymptotically if for any $\xi_i(0)$, $i \in \mathcal{I}$, $\|\xi_i(t) - \xi_j(t)\| \to 0$ as $t \to \infty$ for each $(i,j) \in \mathcal{I}$.

2.2.2 Continuous-time Consensus

The following theorem is our main technical result.

Theorem 2.1. *Given switching interaction topologies and zero transmission or communication noise, the Kalman consensus scheme given in Eqs. (8)–(10) achieves asymptotic consensus if there exist infinitely many consecutive uniformly bounded time intervals such that the union of the interaction graph across each interval has a spanning tree.*

The proof of this theorem depends upon the following five lemmas.

Lemma 2.2. *Let $C(t) = [c_{ij}(t)] \in M_n(\mathbb{R})$ be piecewise continuous, where $c_{ij} \geq 0$, $i \neq j$, and $\sum_j c_{ij} = 0$. Let $\Phi_C(t, t_0)$ be the corresponding transition matrix. Then $\Phi_C(t, t_0)$ is a stochastic matrix with positive diagonal entries for any $t \geq t_0$.*

Proof: From [51], we know that

$$\Phi_C(t,t_0)$$
$$= I + \int_{t_0}^{t} C(\sigma_1)\,d\sigma_1 + \int_{t_0}^{t} C(\sigma_1)\int_{t_0}^{\sigma_1} C(\sigma_2)\,d\sigma_2\,d\sigma_1 + \cdots. \tag{13}$$

Noting that $C(t)\mathbf{1} = 0$, where $\mathbf{1}$ is a column vector of ones, we can verify that $\Phi_C(t,t_0)\mathbf{1} = \mathbf{1}$.

Note that $C(t)$ can be written as $B(t) - \mu I_n$, where $B(t)$ is a nonnegative matrix and μ is a constant greater than $\max_{\tau\in[t_0,t]}\max_{i\in\mathcal{I}}|c_{ii}(\tau)|$. It is straightforward to see that

$$\frac{d}{dt}\Phi_C(t,t_0) = C(t)\Phi_C(t,t_0)$$

and

$$\begin{aligned}
&\frac{d}{dt}[\Phi_B(t,t_0)e^{-\mu(t-t_0)}] \\
&= B(t)\Phi_B(t,t_0)e^{-\mu(t-t_0)} - \mu\Phi_B(t,t_0)e^{-\mu(t-t_0)} \\
&= (B(t)-\mu I_n)\Phi_B(t,t_0)e^{-\mu(t-t_0)} \\
&= C(t)\Phi_B(t,t_0)e^{-\mu(t-t_0)},
\end{aligned}$$

and that $\Phi_C(t_0,t_0) = \Phi_B(t_0,t_0)e^{-\mu(t_0-t_0)} = I$. Therefore, we obtain $\Phi_C(t,t_0) = \Phi_B(t,t_0)e^{-\mu(t-t_0)}$. From Eq. (13), it is straightforward to see that $\Phi_B(t,t_0)$ is nonnegative and has positive diagonal entries. Therefore, it follows that $\Phi_C(t,t_0)$ is nonnegative and has positive diagonal entries. Combining these arguments implies that the transition matrix $\Phi_C(t,t_0)$ is a stochastic matrix with positive diagonal entries. ■

Lemma 2.3. *Let $C(t) = [c_{ij}(t)] \in M_n(\mathbb{R})$ and $\tilde{C} = [\tilde{c}_{ij}(t)] \in M_n(\mathbb{R})$ be continuous on $t \in [\tau,s]$, where $s > \tau$ such that $c_{ij}(t) \geq 0$ and $\tilde{c}_{ij}(t) \geq 0$, $\forall i \neq j$, and $\sum_{j=1}^{n} c_{ij}(t) = \sum_{j=1}^{n}\tilde{c}_{ij}(t) = 0$. Let $\Phi_C(s,\tau)$ and $\Phi_{\tilde{C}}(s,\tau)$ be the corresponding transition matrices. Also let the graph associated with $C(t)$ be fixed for $t \in [\tau,s]$ and suppose that $\tilde{C}(t)$ corresponds to the same fixed graph as $C(t)$. Then the graph of $C(t)$ is a subset of the graph of $\Phi_C(s,\tau)$ and $\Phi_C(s,\tau) \sim \Phi_{\tilde{C}}(s,\tau)$.*

Proof: Let $C(t) = B(t) - \mu I_n$, where $B(t)$ is a nonnegative matrix and μ is a constant greater than $\max_{t\in[\tau,s]}\max_{i\in\mathcal{I}}|c_{ii}(t)|$. Following Lemma 2.2, we know that $\Phi_C(s,\tau) = \Phi_B(s,\tau)e^{-\mu(s-\tau)}$. Note that the graphs associated with $C(t)$ and $B(t)$ are the same, so are the graphs associated with $\Phi_C(s,\tau)$ and $\Phi_B(s,\tau)$. Therefore from Eq. (13), we can see that $\Phi_B(s,\tau) \geq \int_\tau^s B(\sigma_1)d\sigma_1$, where $\int_\tau^s B(\sigma_1)d\sigma_1 \sim B(t)$ for $t \in [\tau,s]$, or in other words, the graph associated with $B(t)$ for $t \in [\tau,s]$ is a subset of the graph associated with $\Phi_B(s,\tau)$. Therefore, the graph associated with $C(t)$ for $t \in [\tau,s]$ is a subset of the graph associated with $\Phi_C(s,\tau)$.

Note that $\Phi_{\tilde{C}}(s,\tau) = \Phi_{\tilde{B}}(s,\tau)e^{-\tilde{\mu}(s-\tau)}$, where $\tilde{C} = \tilde{B} - \tilde{\mu}I_n$. In order to show that Φ_C is of the same type as $\Phi_{\tilde{C}}$, we need to show that Φ_B is of the same type as $\Phi_{\tilde{B}}$. Note that B and $\tilde{B}$ are of the same type since they correspond to the same graph. By writing Φ_B and $\Phi_{\tilde{B}}$ as in Eq. (13) and comparing each term, Lemma 2.1 implies that each corresponding term is of the same type, which in turn implies that $\Phi_B(s,\tau)$ and $\Phi_{\tilde{B}}(\tilde{s},\tilde{\tau})$ are of the same type. ■

Lemma 2.4. *Let $S_A = \{A_1, A_2, \cdots, A_\ell\}$ be a set of stochastic matrices with positive diagonal entries. If the graph associated with A_i has a spanning tree, then A_i is SIA. If the union of the graphs of matrices A_i, $i = 1,\cdots,\ell$, has a spanning tree, then the matrix product $\Pi_{i=1}^{\ell}A_i$ is SIA.*

Proof: The first statement is shown in Corollary 3.5 and Lemma 3.7 in [38]. For the second statement, note that the product of stochastic matrices is still a stochastic matrix. Also note that $\Pi_{i=1}^{\ell}A_i \geq \gamma\sum_{i=1}^{\ell}A_i$ for some $\gamma > 0$ according to Lemma 2 in [34]. Since the union of the graphs of matrices in S_A has a spanning tree, it is obvious that the graph associate with $\sum_{i=1}^{\ell}A_i$ has a spanning tree. Therefore, it can be seen that the graph associated with the matrix product has a spanning tree, which in turn implies, from the first statement of the Lemma, that the matrix product is SIA. ■

Lemma 2.5. *Let $C(t) = [c_{ij}(t)] \in M_n(\mathbb{R})$ be piecewise continuous for $t \in [\tau, s]$, where $s > \tau$ is bounded, $c_{ij} \geq 0$, $i \neq j$, and $\sum_j c_{ij} = 0$. If the union of the directed graphs of matrix $C(t)$ for $t \in [\tau, s]$ has a spanning tree, then the transition matrix $\Phi_C(s, \tau)$ is SIA.*

Proof: Note that $\Phi_C(s,\tau) = \Phi_C(s, t_\ell)\Phi_C(t_\ell, t_{\ell-1}) \cdots \Phi_C(t_1, \tau)$, where t_j, $j = 1, \cdots, \ell$, denotes the times when $C(t)$ is discontinuous. From Lemma 2.3, we know that the graph associated with $C(t)$ for each $t \in [t_{i-1}, t_i]$ is a subset of the graph associated with $\Phi_C(t_i, t_{i-1})$, $i = 1, \cdots, \ell + 1$. In other words, if the union of the directed graphs of matrix $C(t)$ has a spanning tree, so does the union of the directed graphs of the corresponding transition matrices. Also note from Lemma 2.3 that each $\Phi_C(t_i, t_{i-1})$ is a stochastic matrix with positive diagonal entries. The proof then follows from Lemma 2.4. ∎

Before moving on, we need the following definition from [50]. Given a stochastic matrix $S = [s_{ij}] \in M_n(\mathbb{R})$, define

$$\lambda(S) = 1 - \min_{i_1, i_2} \sum_j \min(s_{i_1 j}, s_{i_2 j}).$$

Note that $\lambda(S) \leq 1$ for any stochastic matrix S. If $\lambda(S) < 1$, S is called a scrambling matrix. $\lambda(S) = 0$ if and only if the rows of S are identical. The introduction of λ will be useful for the proof of Theorem 2.1.

Lemma 2.6. *(See [50].) Let $\mathcal{S} = \{S_1, S_2, \cdots, S_k\}$ be a finite set of SIA matrices with the property that for each sequence $S_{i_1}, S_{i_2}, \cdots, S_{i_j}$ of positive length, the matrix product $S_{i_j} S_{i_{j-1}} \cdots S_{i_1}$ is SIA. Then for each infinite sequence $S_{i_1}, S_{i_2}, \cdots$ there exists a column vector ν such that*

$$\lim_{j \to \infty} S_{i_j} S_{i_{j-1}} \cdots S_{i_1} = \mathbf{1}\nu^T. \tag{14}$$

In addition, in the case that $\mathcal{S}$ is an infinite set, $\lambda(W) < 1$, where $W = S_{k_1} S_{k_2} \cdots S_{k_{N_t+1}}$ and N_t is defined as the number of different types of all $n \times n$ SIA matrices. Furthermore, if there exists a constant $0 \leq d < 1$ satisfying $\lambda(W) \leq d$, then Eq. (14) also holds.

Proof: See Lemma 4 and the concluding remarks in [50]. ∎

Proof of Theorem 2.1:

From Eq. (8), we see that $P_i > 0$ is uniformly lower bounded since Q_i is uniformly lower bounded. Also noting that $-P_i[\sum_j g_{ij}(t)(P_j + \Omega_{ij})^{-1}]P_i \leq -P_i^2/(P_i + \Omega_{ii})$, we know that P_i is uniformly upper bounded. From Eq. (9), we can see that $K_{ij}(t) > 0$, $\forall i \neq j$, is uniformly lower and upper bounded.

Let $t_0, t_1, \cdots$ be an infinite time sequence corresponding to the times at which graph $\mathcal{G}(t)$ switches topology. Since the interaction topology cannot switch infinitely fast, we assume that $t_i - t_{i-1} \geq t_L$, $\forall i = 1, 2, \cdots$. Note that each interval $[t_{i-1}, t_i)$ can be divided into finite or infinite number of subintervals such that the length of each subinterval is greater than or equal to t_L but less than or equal to $t_M = 2t_L$ and the graph on each subinterval is fixed. Relabel these subintervals as $s_0, s_1, \cdots$.

Without transmission or communication noise, Eq. (10) can be rewritten in matrix form as $\dot{\xi} = \Lambda(t)\xi$, where $\xi = [\xi_1, \cdots, \xi_n]^T$ and $\Lambda(t) = [\lambda_{ij}(t)]$. As mentioned above, the solution can be denoted as $\xi(t) = \Phi(t, s_j)\Phi(s_j, s_{j-1}) \cdots \Phi(s_1, s_0)\xi(s_0)$, where Φ is the transition matrix. Noting that $\lambda_{ij}(t) = g_{ij}(t)K_{ij}(t)$, $\forall j \neq i$, and $\sum_j \lambda_{ij}(t) = 0$, we know that $\Lambda(t)$ is continuous and satisfies the hypothesis of Lemma 2.3 for $t \in [s_{j-1}, s_j]$. Noting that $K_{ij}(t)$ is uniformly lower and upper bounded, we know that each nonzero, that is, positive, entry λ_{ij}, where $i \neq j$, satisfies the property that $\lambda_{ij} \in [\lambda_L, \lambda_M]$, which is a compact set. In addition, $\lambda_{ii} = -\sum_{j \neq i} \lambda_{ij}$, which is also in a compact set. In the case that the interaction topology is switching with time, there are a finite number of possible interaction topologies. For each possible interaction topology, note that matrix $\Lambda(t)$ has the same structure in the sense that positive, zero, and negative entries are in the same places for $t \in [s_{j-1}, s_j]$. From Lemma 2.3, each transition matrix $\Phi(s_j, s_{j-1})$ is a stochastic matrix, where $t_L \leq s_j - s_{j-1} \leq t_M$, and $\Phi(s_j, s_{j-1})$ is of constant type over this interval, for each possible interaction topology. Combining the above arguments with the fact that $\Phi(s_j, s_{j-1})$ is a continuous function of $\lambda_{ij}(t)$ for $t \in [s_{j-1}, s_j]$, we see that each nonzero entry of $\Phi(s_j, s_{j-1})$ is lower bounded for each possible interaction topology. It is straightforward to see that there are only finitely many types for $\Phi(s_j, s_{j-1})$. We know that there exists a sequence of unions of the directed interaction graphs across some time intervals and each union is uniformly bounded and has a spanning tree. Thus the transition matrix $\Phi^{(k)}$ for each union is a product of finitely many matrices $\Phi(s_{k_i}, s_{k_{i-1}})$. From Lemma 2.1, the type of $\Phi^{(k)}$ is uniquely decided by the order and type of each element in its product. Also, from Lemma 2.5, we know that each $\Phi^{(k)}$ is SIA. In addition, noting that the graph associated with each $\Phi^{(k)}$ has a spanning tree, we see that any number of products of $\Phi^{(k)}$ is also SIA according to the second part of Lemma 2.4. Noting that $\Phi^{(k)}$ can only have finitely many types, we see that for each type of $\Phi^{(k)}$ its nonzero entries are lower bounded. Let $W = \Phi^{(j_1)}\Phi^{(j_2)}\dots\Phi^{(j_{N_t+1})}$. From the second part of Lemma 2.6, we know that $\lambda(W) < 1$. Note that W can only have finite many types, denoted as W_t. In order to show that $\lambda(W) \leq d < 1$, it is sufficient to show that for each type, there exists a $0 \leq d_i < 1$ such that $\lambda(W) \leq d_i$. This can be verified by noting that the nonzero entries of W are lower bounded for each type. Let $d = \max\{d_1, d_2, \cdots, d_{W_t}\}$. It is obvious that $\lambda(W) \leq d$. From Lemma 2.6, we can show that $\xi(t) \to \mathbf{1}\nu^T\xi(0)$, where ν is a nonnegative column vector. ■

In previous results on consensus [34, 38], the coefficient matrix $C(t)$ was assumed to be piecewise constant with finite dwell time, and elements drawn from a finite set. The following corollary of Theorem 2.1 shows that these conditions can be relaxed.

Corollary 2.2. *Let $\dot{\xi} = C(t)\xi$, where $C(t) = [c_{ij}(t)] \in M_n(\mathbb{R})$ is piecewise continuous, $c_{ij} \geq 0$, $i \neq j$, $\sum_j c_{ij} = 0$, and each nonzero entry c_{ij}, $i \neq j$, is both uniformly lower and upper bounded. Under switching interaction topologies, ξ_i achieves consensus if there exist infinite many consecutive uniformly bounded time intervals such that the union of the interaction graph across each such interval has a spanning tree.*

2.2.3 Discrete-time Consensus

Theorem 2.3. *Given switching interaction topologies and zero transmission or communication noise, the discrete-time Kalman consensus scheme listed in Eq. (11)–(12) achieves asymptotic consensus if there exist infinitely many consecutive uniformly bounded time intervals such that the union of the interaction graph across each interval has a spanning tree.*

Proof: Without transmission or communication noise, Eq. (12) can be written as

$$\begin{aligned}\xi_i[k+1] = &\left[1 - P_i[k+1]\sum_{j\neq i} g_{ij}[k](P_j[k]+\Omega_{ij})^{-1}\right]\xi_i[k] \\ &+ P_i[k+1]\sum_{j\neq i}\left[g_{ij}[k](P_j[k]+\Omega_{ij})^{-1}\xi_j[k]\right].\end{aligned} \tag{15}$$

Note that each weighting factor of ξ_ℓ is less than or equal to 1 and the sum of the weighting factors of ξ_ℓ is equal to 1, where $\ell \in \mathcal{I}$. Letting $\xi = [\xi_1, \cdots, \xi_n]^T$, we can rewrite Eq. (15) as $\xi[k+1] = D[k]\xi[k]$, where it can be verified that $D[k]$ is a stochastic matrix with positive diagonal entries. In addition, for each possible interaction topology, $D[k]$ is of the same type and its nonzero entries are lower bounded.

We know that there exists a sequence of unions of the directed interaction graphs across some time intervals and each union is uniformly bounded and has a spanning tree. Let $D^{(i)}$ be the product of matrices $D[k]$ over the ith union. Note that each $D^{(i)}$ is SIA from Lemma 2.4. As a result, the proof follows the same reasoning as the proof of Theorem 2.1 with $D^{(i)}$ playing the role of $\Phi^{(k)}$. ■

2.3 Consensus Schemes are Input-to-State Stable

We are primarily interested in the application of consensus algorithms to cooperative control problems. In this section we will explore a control architecture where a consensus algorithm is in cascade with a coordination algorithm, as shown in Figure 15. Our purpose in this section is to derive conditions on the consensus and coordination algorithms that guarantee that the cooperation objective is achieved. Toward that end, rewrite Eq. (10) as

$$\dot{\xi}_i = \sum_{j=1}^{n} g_{ij}(t)K_{ij}(\xi_j - \xi_i) + \sum_{j=1}^{n} g_{ij}(t)K_{ij}\nu_{ij}. \tag{16}$$

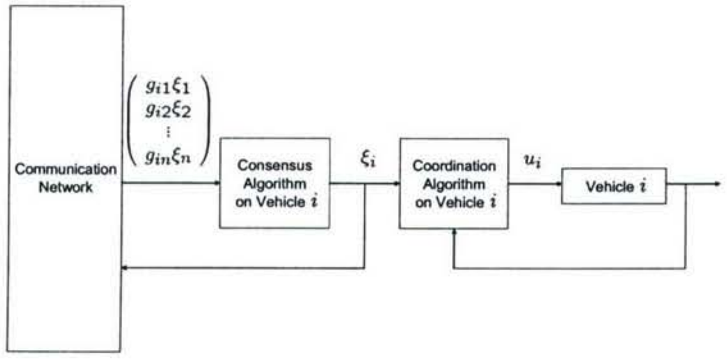

Communication Network
$\begin{pmatrix} g_{i1}\xi_1 \\ g_{i2}\xi_2 \\ \vdots \\ g_{in}\xi_n \end{pmatrix}$
Consensus Algorithm on Vehicle i
ξ_i
Coordination Algorithm on Vehicle i
u_i
Vehicle i

Figure 15: The control architecture consists of a consensus algorithm in cascade with a coordination algorithm. The consensus algorithm receives information from the communication network to produce a value of the coordination variable ξ_i. The coordination algorithm uses the coordination variable ξ_i to produce a command to the vehicle u_i. We assume that identical consensus and coordination algorithms are implemented on each vehicle.

Letting $x_{ij} = \xi_i - \xi_j$ and $x = (x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, \ldots, x_{nn})^T$, we get the state-space model

$$\dot{x} = A(t)x + B(t)\nu \tag{17}$$

where ν is a column vector created by stacking the communication noise terms ν_{ij}, and the elements of $A(t)$ and $B(t)$ are linear combinations of $g_{ij}K_{ij}(t)$ and can be easily constructed from Eq. (16). The vector x represents the total consensus error.

Theorem 2.4. *Under the hypothesis of Theorem 2.1, the Kalman consensus scheme given by Eqs.* (8), (9), *and* (17) *is input-to-state stable.*

The proof of this theorem requires the following two lemmas.

Lemma 2.7. *Under the hypothesis of Theorem 2.1, if the communication error ν is zero, then the consensus error x is uniformly stable.*

Proof: Note that $\xi(t) = \Phi(t,t_0)\xi(t_0)$, where $\Phi(t,t_0)$ is a stochastic matrix according to Lemma 2.2. As a result, we see that the i^{th} coordination variable $\xi_i(t)$ is equal to a weighted average of all agents' initial coordination variables communicating with agent i. Since a weighted average can never be greater (or smaller) than any one of the components in the average, we know that $\xi_i(t) \in [\min_j \xi_j(t_0), \max_j \xi_j(t_0)]$ for all t and i. Then it is straightforward to see that

$$\|x(t)\|_\infty \leq \|x(t_0)\|_\infty, \quad \text{for } t \geq t_0.$$

■

Lemma 2.8. *The norm of $B(t)$ in Eq.* (17) *is bounded.*

Proof: Since $B(t)$ is composed of linear combinations of $K_{ij}(t)$, if $\|K_{ij}(t)\|$ is bounded for each (i,j), then $\|B(t)\|$ will also be bounded. $\|K_{ij}\|$ was shown to be bounded in the proof of Theorem 2.1. ■

Proof of Theorem 2.4: By Lemma 2.7, the Kalman consensus error is uniformly stable. By Theorem 2.1, $\|\xi_i - \xi_j\| \to 0$ as $t \to \infty$ for all (i,j). Since each element of $x \to 0$, then $\|x\| \to 0$ as $t \to \infty$ and we conclude uniform asymptotic stability. Any linear system that is uniformly asymptotically stable is also uniformly exponentially stable [51]. Additionally, linear uniformly exponentially stable systems with $\|B(t)\| < \beta$ for finite β are bounded-input bounded-output stable [51]. Since the Kalman consensus error governed by Eq. (17) is a linear uniformly asymptotically stable system with $\|B(t)\|$ bounded, it is ISS. ■

Corollary 2.5. *If the continuous-time consensus schemes presented in [16], [38], [34], and [35] are augmented with communication noise, then the representation of these schemes that is equivalent to Eq.* (17) *is ISS.*

Proof: The difference between each of these schemes and Eq. 10 is that the consensus gain $K_{ij}(t)$ is time invariant. Therefore, from the proof of Theorem 2.4 it is clear that they are ISS. ■

Referring to Figure 15 we see that the combination of the communication network and the consensus scheme is an ISS system. It is well known that the cascade combination of two ISS systems is also ISS. Therefore if the feedback loop containing the coordination algorithm and the i^{th} vehicle is ISS from the consensus error to the cooperation objective, then the total system will be ISS from the communication noise to the cooperation objection. This concept is shown schematically in Figure 16 and can be summarized by the following Theorem.

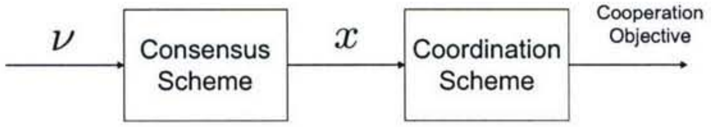


Figure 16: The distributed cooperative control problem can be thought of as a cascade connection between the consensus algorithm and the coordination algorithm. If both are ISS, then the cascade system will be ISS

Theorem 2.6. *If a consensus scheme is ISS from the communication noise to the consensus error and a coordination scheme is ISS from the consensus error to the cooperation objective, then the cascade interconnection of the two (see Fig. 16) is ISS from the communication noise to the cooperation objective.*

2.4 Illustrative Example - Distributed Cooperative Timing for a Team of UAVs

Suppose that a team of UAVs, flying at distinct altitudes, is tasked to simultaneously visit a pre-specified location. For simplicity, also assume that paths with appropriate velocities have been precomputed for each UAV as shown in Figure 17. Algorithms that achieve this functionality are described in [41].

We will also assume that each UAV has autopilot functionality that maintains the UAV on its pre-defined path, but that the velocity along the path can be adjusted to meet the simultaneous arrival objective [17]. We will assume that the velocity hold autopilot has been designed such that

$$\dot{v}_i = \alpha_i \left(v_i^c - v_i\right) \tag{18}$$

where $\alpha_i > 0$, v_i is the velocity, and v_i^c is the commanded velocity for the i^{th} UAV. Let L_i denote the length of the path remaining to the target, then

$$\dot{L}_i = -v_i.$$

Given L_i and v_i, the i^{th} UAV can estimate its expected time-of-arrival as

$$\tau_i = \frac{L_i}{v_i}.$$

Therefore

$$\begin{aligned}\dot{\tau}_i &= \frac{v_i\dot{L}_i - L_i\dot{v}_i}{v_i^2}\\ &= -1 - \alpha_i\tau_i\left(\frac{v_i^c - v_i}{v_i}\right).\end{aligned}$$

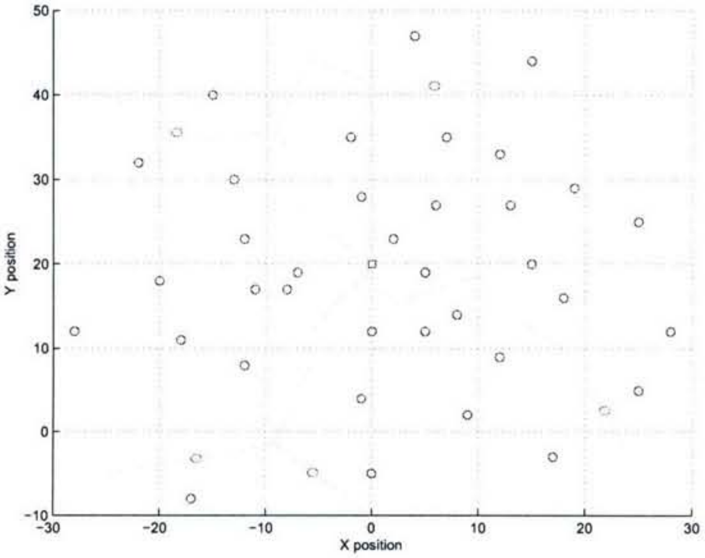


Figure 17: Cooperative timing scenario with five UAVs.

The cooperation objective for this problem is that each UAV arrives at its destination simultaneously, i.e. $\tau_i - \tau_j = 0$ for each (i, j). The coordination variable for this problem is chosen as the arrival time. Therefore ξ_i represents the i^{th} UAVs understanding of the team arrival time. Letting

$$v_i^c = v_i + \frac{v_i}{\alpha_i \tau_i}\left(\gamma\tau_i - \gamma\xi_i - 1\right) \tag{19}$$

we get that

$$\dot{\tau}_i = -\gamma\tau_i + \gamma\xi_i.$$

Note that

$$\begin{aligned}(\dot{\tau}_i - \dot{\tau}_j) &= -\gamma\tau_i + \gamma\xi_i + \gamma\tau_j - \gamma\xi_j \\ &= -\gamma\left(\tau_i - \tau_j\right) + \gamma\left(\xi_i - \xi_j\right),\end{aligned}$$

and that the system $\dot{\phi} = -\gamma\phi + \gamma u$ is input-to-state stable. In fact we have that

$$|\phi(t)| \le e^{-\gamma(t-t_0)}\phi(t_0) + \sup_{t_0 \le \sigma \le t} |u(\sigma)|.$$

Therefore, from Theorem 2.6, the combination of the consensus strategy given by Eqs. (8)–(10) and the velocity controller given by Eq. (19) is input-to-state stable with the input being communication noise and the state consisting of both the consensus discrepancy $\xi_i - \xi_j$ and the UAV arrival discrepancy $\tau_i - \tau_j$.

The cooperative timing scenario was simulated with an unreliable switching communication topology. The team is connected in the graph shown in Fig. 18 where each link is only available 70 percent of the time. When an agent receives communication it updates its estimate of ξ using the Kalman consensus scheme of Section 2.1.1. In between consensus updates, agents control their velocity using Equation (19). Five agents were given a single target at which to arrive simultaneously. Fig. 17 shows the application scenario, where each red circle represents an agent, the blue circles represent threats, the blue square represents the target, and the green lines are the waypoint paths.

In the first case, communication noise was set to zero and each agent started with approximately the same confidence in its estimate of ξ. A plot of ξ for each vehicle is shown in Fig. 19(a) and Fig. 19(b) shows

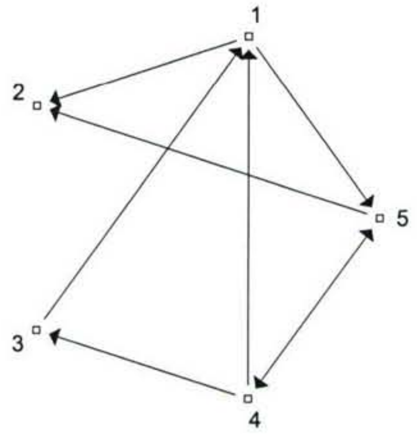
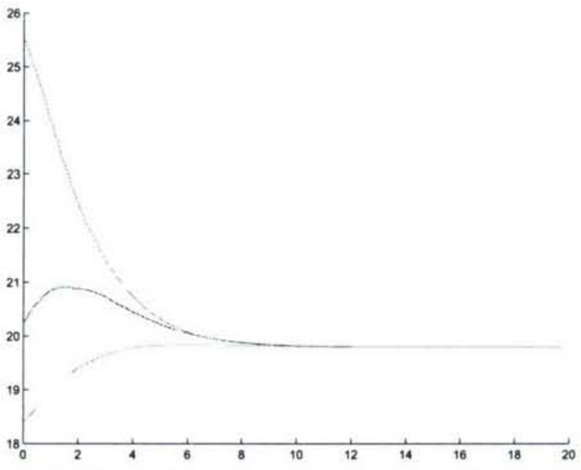


Figure 18: Union of possible communication topologies.

τ for each vehicle. As can be seen, each agent in the team achieves agreement using consensus, adjusts its velocity to match ξ_i, and arrives at the target in approximately 20 seconds.


(a) Estimated team time of arrival, ξ, for each agent

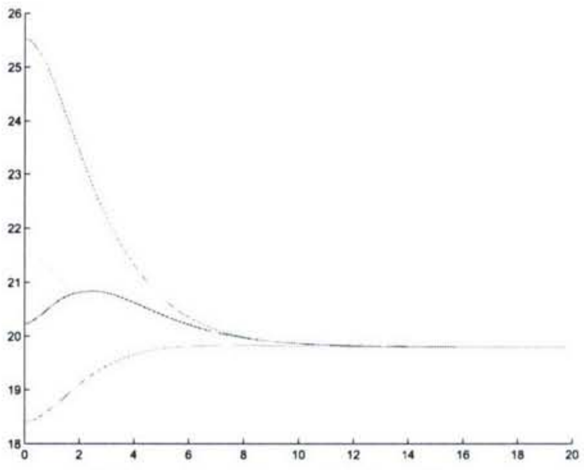
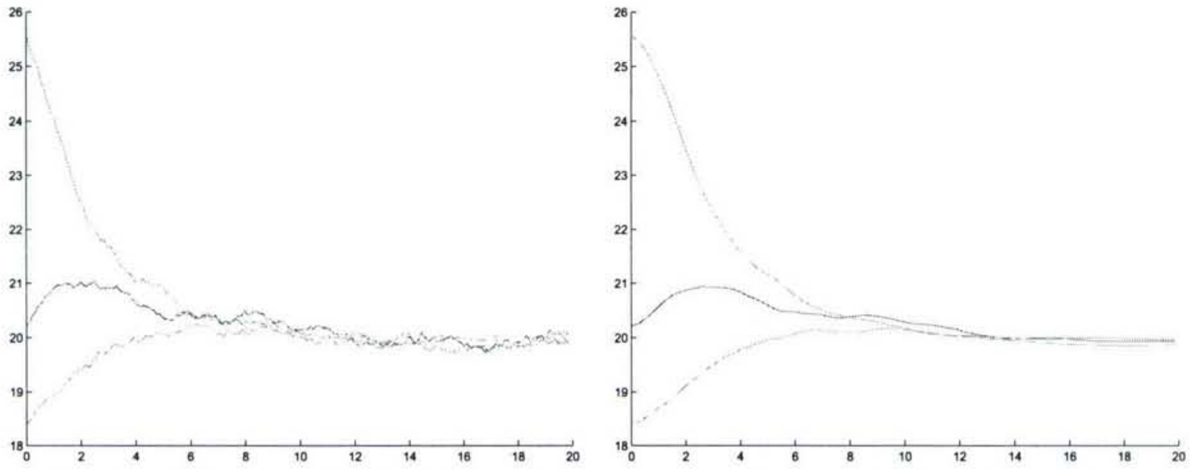

(b) Actual time of arrival, τ, for each agent

Figure 19: Cooperative timing with no communication noise.

In the second case, significant communication noise is added. ξ is shown for each vehicle in Fig. 20(a) and τ for each vehicle is shown in Fig. 20(b). As can be seen, each agent in the team achieves approximate agreement using consensus where the error in agreement is due to the communication noise.

2.5 Conclusions

This section has considered the problem of consensus seeking with relative uncertainty in distributed multi-agent systems. We have proposed discrete-time and continuous-time Kalman filter-like consensus schemes that are appropriate when different agents in the group may have different confidences about their information state. Sufficient conditions have been shown for consensus seeking using the proposed consensus schemes



(a) Estimated team time of arrival, ξ, for each agent

(b) Actual time of arrival, τ, for each agent

Figure 20: Cooperative timing with significant communication noise.

under switching interaction topologies. Consensus schemes were shown to be input-to-state stable from the communication noise to the consensus error. This fact was exploited in an application to a UAV distributed cooperative timing scenario.

3 Average Consensus and Message Passing

For teams of autonomous agents, the ability to cooperate in a decentralized manner can enhance the overall effectiveness of the team. Central to decentralized cooperation is the consensus problem which has been investigated recently by a number of researchers [34, 19, 35, 36].

In general terms, the consensus problem for a group of agents is to ensure that as time progresses each agent approaches a consistent understanding of their shared information. In the general problem, the value to which the team converges is arbitrary, the only requirement being that all agents eventually agree. Average-consensus problems add the restriction of requiring that the final value (the group decision) be the exact average of the agents' initial values. The average-consensus problem is a part of the family of χ-consensus problems [52] where the value that the team is to converge is a function χ of the initial values of the team (e.g. max or min). Recently average-consensus has been used as a basis for distributed Kalman filters [53, 54].

In [35] Olfati-Saber and Murray propose a distributed, linear, continuous-time protocol that ensures that average-consensus is achieved asymptotically if the interaction networks connecting the agents switch between balanced, strongly connected graphs. This section will extend those results to the discrete-time domain as well as relax the restriction of requiring the interaction topology to be strongly connected at each instant. Our main result will be to show that if the interaction topology at any instant is balanced and the *union* of the network graph is strongly connected over every interval T, then average-consensus is still achieved asymptotically. Thus, a network may at no instant be strongly connected, yet agents in a team can still achieve average-consensus.

This section is outlined as follows. In Section 3.1 we introduce a formal definition of average-consensus as well as notation that relates the communication topology to consensus protocols. Our main results are presented in Section 3.2. Section 3.3 investigates the practical issues in forming an average-consensus protocol in the discrete-time framework and proposes two such protocols. The notion of *deadbeat* consensus is introduced in Section 3.4 and Section 3.5 investigates trade-offs between asymptotic and finite-time average-consensus protocols. Finally, conclusions are offered in Section 3.6.

3.1 Definitions and Terminology

The information flow topology between agents on a team is most naturally represented as a directed graph. For this reason, we introduce graph theoretic terminology similar to [48].

Let $A = [a_{ij}]$ be an $n \times n$ nonnegative matrix. The underlying directed graph G associated with A has vertex set $V(G) = \{1, \dots, n\}$ and a directed edge (i, j) from node i to j if and only if $a_{ji} \neq 0$ (note: some authors use the transpose of A, i.e. there is a directed edge (i, j) from i to j if and only if $a_{ij} \neq 0$). As we have defined the relationship between a matrix and its underlying graph, the nodes sending information to node i can be determined by the nonzero entries in row i. Nonzero entries in column i indicate which nodes are receiving information from node i. Note that two matrices with nonzero entries in the same locations have the same underlying graph. The neighbors, N_i, of node i are all nodes that communicate to i, i.e. $N_i = \{j \mid a_{ij} \neq 0\}$. By convention, we assume that each node can communicate with itself, so $a_{ii} > 0\ \forall i$ and $i \in N_i$.

The graphs associated with matrices can be connected in a variety of ways. Connectivity of the network can be roughly classified as follows:

- *Fully Connected:* Each node has as its neighbors all other nodes in the network.
- *Strongly Connected:* Each node has a path that follows the directed edges of the graph to every other node in the network. A direct connection to all other nodes is not necessary, but information flow from each node must reach all other nodes.
- *Spanning Tree:* At least one node has a path that follows the directed edges of the graph to every other node in the network.

Graphs can also be connected over time by considering the union of the communication links over an interval of time (i.e. the union contains all edges that were active during that interval). A reversed graph is simply a graph with the direction of the links reversed. Note that a reversed graph is associated with the transpose of the original matrix.

Each node has an associated value $x_i \in \mathbb{R}$ which represents the information on which the team must come to agreement. The set of nodes $\{1, \ldots, n\}$ is said to be in consensus if $x_i = x_j$ for all i, j. When each $x_i = \frac{1}{n}\sum_j x_j[0]$ the team is said to have reached average-consensus. A consensus protocol defines how a node should update is value of x_i based on the values of its neighbors. The simplest scheme is to require that each node update its value x_i to some weighted linear combination of its neighbors values.

$$x_i[k+1] = \sum_{j \in N_i} a_{ij} x_j[k]$$

The dynamics of the information vector $x = \{x_1, \ldots, x_n\}$ can then be defined as

$$x[k+1] = A[k]x[k]$$

where the *sign* of each entry in $A[k]$ is given by the communication topology at time k, but the value a_{ij} for the nonzero elements is determined by the protocol.

Let $\Phi_A(k, k_0) = A[k]A[k-1]\cdots A[k_0]$, then at each k the information vector can be described by

$$x[k+1] = \Phi_A(k, 0)x[0].$$

Consensus is said to be reached asymptotically if

$$\lim_{k \to \infty} \Phi_A(k, 0) = \mathbf{1}y^T \tag{20}$$

where $\mathbf{1}$ is the vector of all ones, $y_i \geq 0$, and $\mathbf{1}^T y = 1$. Notice that if Eq. (20) is satisfied, then $x \to \mathbf{1}y^T x[0]$ implying that each x_i approaches the same convex combination of the agents' initial values. Equivalently, average-consensus is said to be reached asymptotically if

$$\lim_{k \to \infty} \Phi_A(k, 0) = \frac{1}{n}\mathbf{1}\mathbf{1}^T. \tag{21}$$

3.2 Average-Consensus under Switching Topologies

The results for linear consensus protocols under switching interaction topologies have been well studied [19, 36] with the main result being that the union of the interaction graphs over every interval T must contain a spanning tree to reach consensus. We will draw similar conclusions with respect to average-consensus. Theorem 1 develops the conditions for each $A[k]$ that allows Eq. (21) to be satisfied. This requires the following two Lemmata.

Lemma 1 (Proposition 1 in [36]). *Let $x[k+1] = A[k]x[k]$ where $A[k] = [a_{ij} \geq 0]$, $\sum_j a_{ij} = 1$, $a_{ii} > 0$ for all k, and each nonzero entry a_{ij} is both uniformly upper and lower bounded. If there exists $T \geq 0$ such that for every interval $[k, k+T]$ the union of the interaction graph across the interval contains a spanning tree, then consensus is asymptotically achieved (i.e. Eq. (20) is satisfied).*

A similar result is implicit in [55]. Notice that each node has the ability to choose the weight associated with the information from each of its neighbors to ensure that its row sums to one. If the team is connected often enough (i.e. has a spanning tree over every interval of length T), then Lemma 1 ensures that consensus is reached.

Lemma 1 requires that the row sums of $A[k]$ be one and that a spanning tree be achieved in every interval of length T for consensus to be reached. Now consider the reversed dynamics $x[k+1] = B[k]x[k]$ where each *column* sum is equal to one.

Lemma 2. *Let $x[k+1] = B[k]x[k]$ where $B[k] = [b_{ij} \geq 0]$, $\sum_i b_{ij} = 1$, $b_{ii} > 0$, and each nonzero entry b_{ij} is both uniformly upper and lower bounded. Under switching interaction topologies, if there exists $T \geq 0$ such that for every interval $[k, k+T]$ the union of the* reverse *interaction graph across the interval contains a spanning tree, then*

$$\lim_{k\to\infty} \Phi_B(k, k_0) = y\mathbf{1}^T$$

where $y_i \geq 0$ and $y^T\mathbf{1} = 1$.

Proof: If the column sums of $B[k]$ are equal to one and a spanning tree is achieved in the reverse graph, then $B^T[k]$ has row sums of one and a spanning tree is achieved in the regular graph. By application of Lemma 1 $\lim_{k\to\infty} \Phi_{B^T}(k, k_0) = \mathbf{1}z^T$, so

$$\begin{aligned}
\lim_{k\to\infty} \Phi_{B^T}(k_0, k) &= \mathbf{1}y^T \\
[\lim_{k\to\infty} \Phi_{B^T}(k_0, k)]^T &= [\mathbf{1}y^T]^T \\
\lim_{k\to\infty} \Phi_{B^T}(k_0, k)^T &= y\mathbf{1}^T \\
\lim_{k\to\infty} \Phi_B(k, k_0) &= y\mathbf{1}^T
\end{aligned}$$

The fact that $\Phi_{B^T}(k, k_0) = \mathbf{1}z^T \Rightarrow \Phi_{B^T}(k_0, k) = \mathbf{1}y^T$ can be seen by noting that each $B^T[k]$ is row stochastic with positive diagonal entries and if the product $B^T[k]B^T[k+1] \cdots B^T[k+T]$ contains a spanning tree, then the product $B^T[k+T]B^T[k+T-1] \cdots B^T[k]$ also contains a spanning tree. Wolfowitz [50] showed that infinite products of SIA matrices (a superset of matrices that have a spanning tree and are row stochastic with positive diagonal entries) converge to the form $\mathbf{1}y^T$ in *any* product order (however, the value of y will be dependent on the actual order). ■

Theorem 1. *Let $x[k+1] = A[k]x[k]$ where $A[k] = [a_{ij} \geq 0]$, $\sum_i a_{ij} = 1$, $\sum_j a_{ij} = 1$, $a_{ii} > 0$, and each nonzero entry a_{ij} is both uniformly upper and lower bounded. Under switching interaction topologies, if there exists $T \geq 0$ such that for every interval $[k, k+T]$ the union of the interaction graph across the interval is strongly connected, then Eq. (21) is satisfied and average-consensus is reached asymptotically.*

Proof: Since $A[k]$ is strongly connected over each interval $[k, k+T]$, then $A[k]$ has a spanning tree in both the regular graph and the reverse graph. Therefore, the matrix $A[k]$ satisfies all the conditions in Lemma 1 *and* Lemma 2. Consequently,

$$\lim_{k\to\infty} \Phi_A(k, k_0) = \mathbf{1}y^T$$

and

$$\lim_{k\to\infty} \Phi_A(k, k_0) = z\mathbf{1}^T$$

so

$$\lim_{k\to\infty} \Phi_A(k, k_0) = \frac{1}{n}\mathbf{1}\mathbf{1}^T.$$

■

3.3 Distributed Protocol

Careful examination of Theorem 1 will reveal that finding a distributed protocol to satisfy the hypotheses of the theorem will be difficult. Specifically, at each instant in time, the row *and* column sums must be equal to one. In the general consensus problem, only the row sums are required to be one. Since the neighbors of agent i are determined completely by row i, then each agent simply chooses appropriate weights for each of its neighbors values ensuring that the weights sum to one. In the average-consensus case, not only do the weights associated with the neighbors of i need to sum to one, but all nodes for which i is a neighbor must weight the information from i such that the column sum is equal to one. This section will investigate this subtlety and propose two protocols that achieve average-consensus in a distributed manner.

To illustrate the difficulty of requiring both row and column sums to be one, consider the network topology shown in Figure 21. The matrix

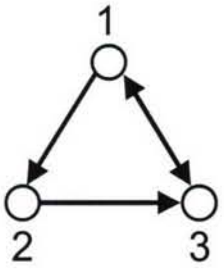


Figure 21: Simple network over which the average-consensus problem can be solved, but which requires global information to be available.

$$A = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 2 & 0 & 1 \end{bmatrix}$$

has as its underlying graph the topology shown in Figure 21 and has row and column sums equal to one. If the underlying network topology remains fixed, then by Theorem 1, the system will achieve average-consensus asymptotically. In one sense, the protocol is distributed since each agent only uses the information received from its neighbors; however, the first agent weights all neighbors' values equally, the second agent weights its own value twice as much as its neighbors, and the third agent weights its neighbors values twice as much as its own. In order to determine the entries in A, some global knowledge of the network topology is required - i.e. there is no simple rule that an agent can use to determine the weight it gives to information from its neighbors without knowledge of the global topology.

An ideal protocol would be able to achieve average-consensus without using global information. We will investigate two protocols that impose additional restrictions on the types of graphs involved, but that achieve average-consensus without resorting to global information. The first is proposed in [35] and requires the definition of the graph Laplacian. Let L be defined element-wise as

$$\ell_{ij} = \begin{cases} \sum_{k=1,k\neq i}^{n} \alpha_{ik}, & j = i \\ -\alpha_{ij}, & j \neq i \end{cases}$$

where $\alpha_{ij} = 1$ if there is a communication link from node j to node i and $\alpha_{ij} = 0$ otherwise (here $\mathcal{A} = [\alpha_{ij}]$ is simply the adjacency matrix of a graph G). The protocol is then defined in terms of the Laplacian

$$x[k+1] = (I - \epsilon L)x[k] \tag{22}$$

where $\epsilon \in (0, 1/\max_i \ell_{ii})$. Notice that the row sums of L are all zero by construction, so the row sums of $A = I - \epsilon L$ are all one. If $\epsilon \in (0, 1/\max_i \ell_{ii})$, then A will also be nonnegative and consensus will be guaranteed if the graph contains a spanning tree in every interval T.

Olfati-Saber and Murray show in [35] that when a graph is *balanced* then the column sums of L are zero. A balanced graph is one in which at each node the out-degree equals the in-degree, i.e. each node sends information to as many as send information to it. Notice that when G is balanced then L has column sums of zero, and A has column sums of one. So, by Theorem 1, the protocol (22) will achieve average-consensus if the network switches between instantaneously balanced networks which are strongly connected over every interval T.

Protocol (22) is almost completely distributed since each node determines the weight to associate with information from its neighbors without knowledge of the graph topology; however, all nodes must have the same value of ϵ whose upper limit is determined by the connectivity of the graph. Certainly, for fixed number of agents n, $\epsilon \in (0, 1/n]$ will ensure that A remain nonnegative and Theorem 1 will apply. This requires *a priori* knowledge of the size of the team, especially since the larger the value of ϵ the faster the rate of convergence (the second eigenvalue will be closer to zero, see [35]). Setting $\epsilon = 1/N$ where N is an upper bound on the number of agents on the team will ensure that average-consensus is achieved asymptotically.

To illustrate the applicability of this protocol consider a scenario where the network topology switches randomly from between the graphs in Figure 22. The convergence for two values of ϵ are shown in Figures 23 and 24. In this example, the sum of the initial conditions is one. Notice that with both values of ϵ the value to which the system converges is $\frac{1}{4}$, but the larger value of ϵ gives faster convergence.

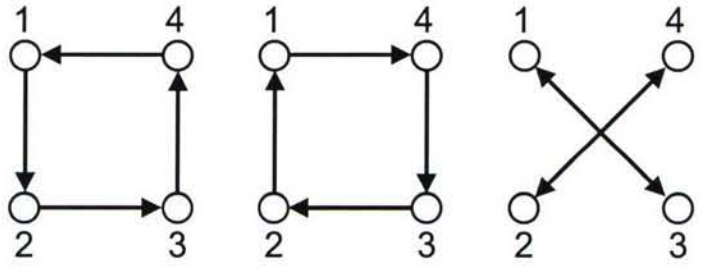


Figure 22: Example scenario where the network topology switches randomly between these three graphs.

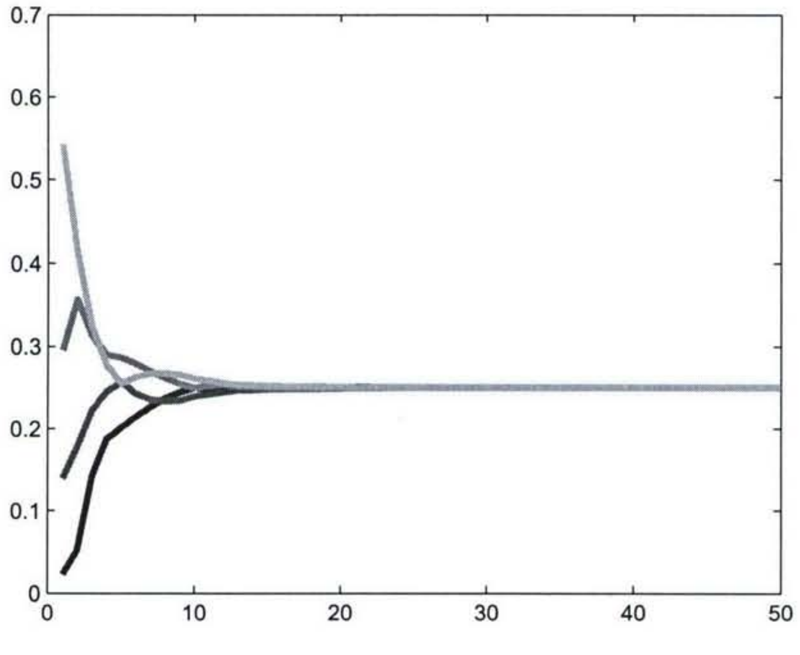


Figure 23: Protocol (22) with $\epsilon = \frac{1}{4}$ under switching topologies.

When every communication link is bi-directional (i.e. G is an undirected graph), then the graph is trivially balanced. In this case, it is possible to develop a protocol that can be implemented without *a priori* knowledge of team size. Assume that agents have the ability to negotiate with each of their neighbors to

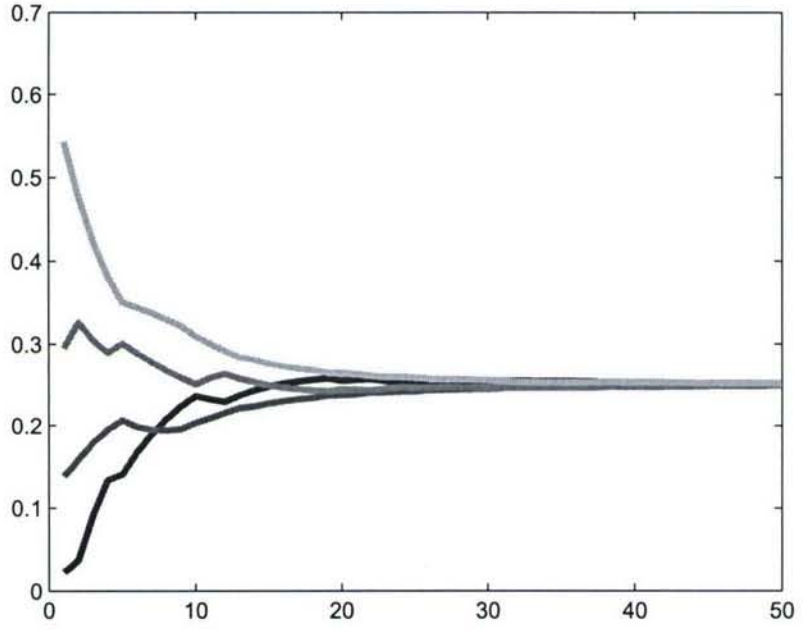


Figure 24: Protocol (22) with $\epsilon = \frac{1}{8}$ under switching topologies.

isolate the exchange of information to just one neighbor at a time. During this communication event, both agents update their values to be the exact average of the values present. For a two-agent communication event (i, j), the protocol matrix A will be the identity matrix with the exception of $a_{ij} = a_{ji} = a_{ii} = a_{jj} = \frac{1}{2}$. Notice that each $A[k]$ retains the characteristic of having row and column sums equal to one. Essentially, each agent cycles through available communication channels to isolate a single neighbor at a time and effectively change its in-degree and out-degree to one at each instant. If over every interval T the union of these simple graphs is connected, then the conditions in Theorem 1 are satisfied and average-consensus is achieved asymptotically.

An example of this protocol is shown in Figure 26. In this scenario, the agents are connected in a static graph of the form $\{1 \leftrightarrow 2, 2 \leftrightarrow 3, 3 \leftrightarrow 4\}$. The agents negotiate with their neighbors so that each agent only communicates with one other agent at a time. For simulation purposes, this can be modeled as the system switching randomly between the graphs in Figure 25. Observe that the final value is the exact average of the agents' initial conditions.

To summarize, the Laplacian protocol of Eq. (22) can achieve average-consensus if the interaction topology is balanced at each instant and is strongly connected over every interval T. It requires that some knowledge of the maximum connectedness or maximum number of agents be available *a priori* to determine the parameter ϵ. A second protocol reduces available communication to allow only simple two-agent interactions. This requires bi-directional communication between agents (more restrictive than balanced) but assumes no *a priori* knowledge of the network topologies or team size.

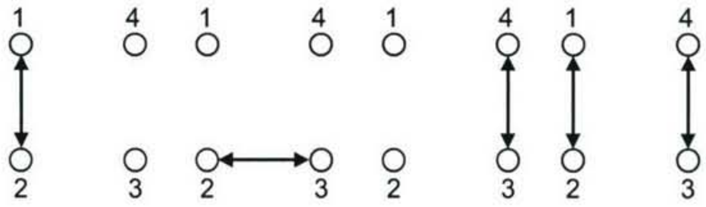


Figure 25: Example scenario where the topology remains fixed (a path), but the agents negotiate through one of the above graphs at each instant.

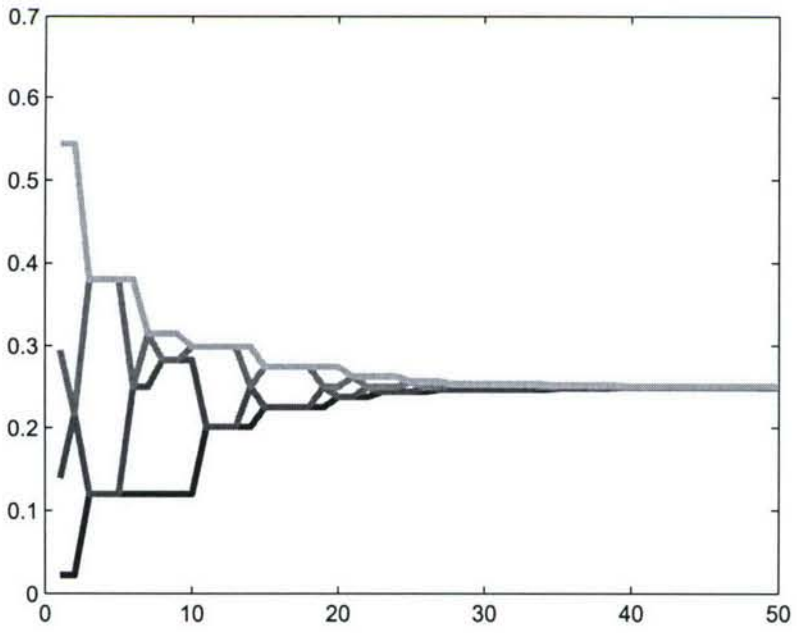


Figure 26: Results of simple two-agent events to achieve average consensus.

3.4 Deadbeat Consensus

Discrete-time systems can exhibit finite-time convergence when the poles of the system are all at zero - i.e. when a system

$$x[k+1] = Ax[k]$$

has nilpotent matrix A, then $x[k] = 0 \ \forall k > n$, regardless of initial conditions. This notion of "deadbeat" response motivates a similar investigation of consensus systems. This section will consider the conditions under which consensus can be reached in finite time.

Let $\mathcal{P}$ be a consensus protocol; specifically, let the interaction matrices generated by $\mathcal{P}$ have row sums equal to one. Note that if $\mathcal{P}$ solves either the general consensus problem or the average-consensus problem, the row sums of the interaction matrices will be one. At any time k, the value of the system given the initial conditions at $k = 0$ is

$$x[k+1] = (A[k]A[k-1] \cdots A[1]A[0])x[0].$$

Theorem 2. *Let $\mathcal{P}$ be a consensus protocol. If at some instant, ℓ, the interaction topology is a fully connected graph and $\mathcal{P}$ yields the interaction matrix at that instant*

$$A[\ell] = \frac{1}{n}\mathbf{1}\mathbf{1}^T$$

then the team will be exactly in consensus for all $k > \ell$.

Proof: A group of agents is in consensus if $x_i = x_j$ for every pair (i, j). Since

$$x[\ell+1] = A[\ell]x[\ell] = \frac{1}{n}\mathbf{1}\mathbf{1}^T x[\ell]$$

then each element i of $x[\ell+1]$ is

$$x_i[\ell+1] = \frac{1}{n}\sum_{j=1}^{n} x_j[\ell].$$

Therefore, the group has reached consensus at time $(\ell+1)$. Because $\mathcal{P}$ ensures that each interaction matrix after time ℓ has row sums equal to one, then for all $k > \ell$ the group remains in consensus (each node updates to a weighted sum of the same value).

To show that the same conditions on A lead to deadbeat average-consensus, notice that $\mathcal{P}$ has row and column sums equal to one at each instant so

$$\sum_{i=1}^{n} x_i[k] = \sum_{i=1}^{n} x_i[0]$$

for all $k \geq 0$. Recall that if a matrix A has row and column sums equal to one, then $\mathbf{1}$ is both a left and right eigenvector associated with eigenvalue 1, so

$$\begin{aligned}\sum_{i=1}^{n} x_i[k] &= \mathbf{1}^T x[k] \\ &= \mathbf{1}^T A[k]x[k-1] \\ &= \mathbf{1}^T x[k-1] \\ &\vdots \\ &= \mathbf{1}^T x[0] \\ &= \sum_{i=1}^{n} x_i[0].\end{aligned}$$

Therefore, for each agent i at time ℓ

$$x_i[\ell+1] = \frac{1}{n}\sum_{j=1}^{n} x_j[\ell] = \frac{1}{n}\sum_{j=1}^{n} x_j[0].$$

which implies that the group has reached average-consensus. ■

Theorem 2 requires that at some instant the communication graph is fully connected, i.e every agent can communicate with every other agent *and* that the interaction matrix generated by the consensus protocol yields $A = \frac{1}{n}\mathbf{1}\mathbf{1}^T$. One consequence of this is that deadbeat average-consensus is much more difficult to achieve than regular deadbeat consensus. This is due to the fact that average-consensus protocols do not generally yield the proper interaction matrix when the communication graph is fully connected.

The reader will note that even when a graph is fully connected neither protocol from Section 3.3 will yield the proper interaction matrix to achieve deadbeat consensus. A consensus protocol of the form

$$x_i[k+1] = \frac{1}{|N_i|} \sum_{j \in N_i} x_j[k] \tag{23}$$

will allow regular deadbeat consensus since whenever the network is fully connected, each agent updates its value to the average of all the other agents. Unfortunately, such a simple protocol does not lead to average-consensus in the general case. Consider the balanced network shown in Figure 27 with interaction matrix

$$A = \frac{1}{6} \begin{bmatrix} 3 & 0 & 3 & 0 \\ 2 & 2 & 0 & 2 \\ 0 & 3 & 3 & 0 \\ 0 & 3 & 0 & 3 \end{bmatrix}$$

which does not have column sums equal to one. In fact

$$\lim_{k \to \infty} A^k = \frac{1}{9} \begin{bmatrix} 2 & 3 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 3 & 2 & 2 \end{bmatrix}$$

which shows that the protocol defined by Eq. (23) is not an average-consensus protocol. It is interesting

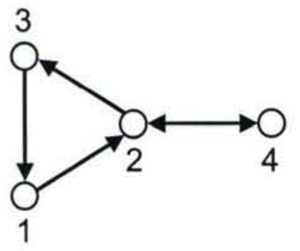


Figure 27: Balanced graph for which a simple averaging protocol does not achieve average-consensus.

to note that if the network topologies switch between balanced *regular* graphs (graphs where all nodes that have any adjacent edge have the same degree), then this protocol does achieve average-consensus (in the general and deadbeat case).

3.4.1 Example Application

Consider a fixed perimeter which is to be monitored by a team of N agents in a distributed manner (as in [56]). Let the consensus variable in the system be the size of the segment an agent is to monitor. The initial state of the system is when the first agent reaches the endpoint of the perimeter and initializes its consensus variable to the length of the perimeter. We desire average-consensus so that asymptotically, each agent monitors an equal part of the perimeter.

Using the protocol of Eq. (23) and noticing that the system will only switch between balanced regular graphs (since agents meet along a line) deadbeat average-consensus may be reached. Figure 28 shows a scenario where at no time are all agents in communication, so average-consensus is achieved asymptotically. In contrast, Figure 29 shows a scenario where agents are launched in close proximity and meet in a fully connected group near the beginning of the mission achieving deadbeat average-consensus.

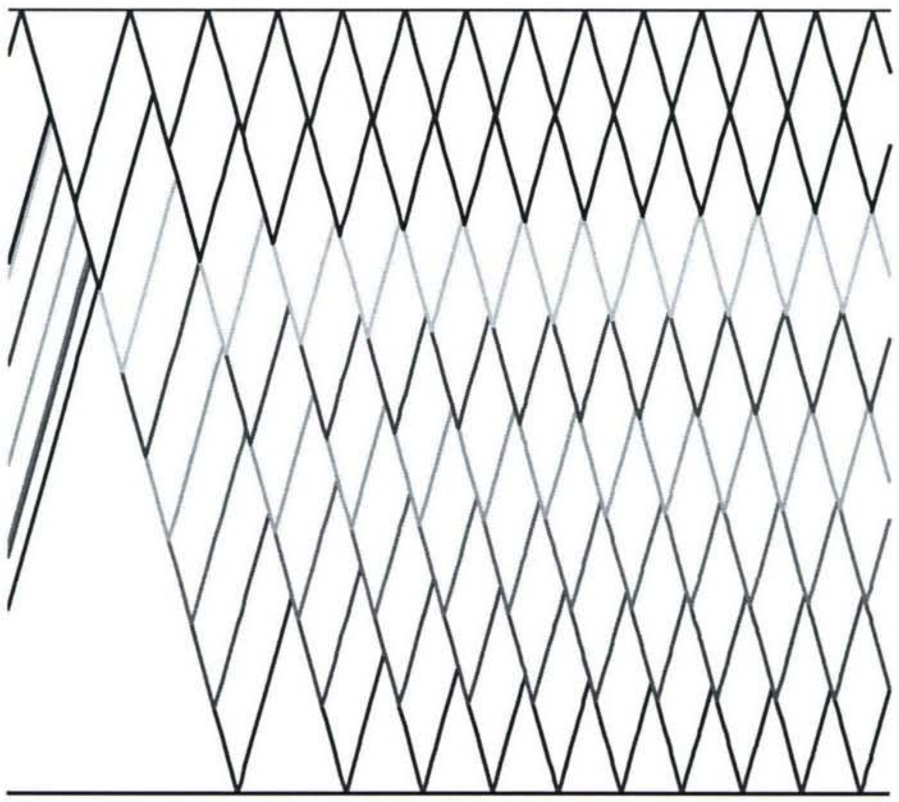


Figure 28: Perimeter surveillance using average-consensus to distribute the team evenly along the perimeter.

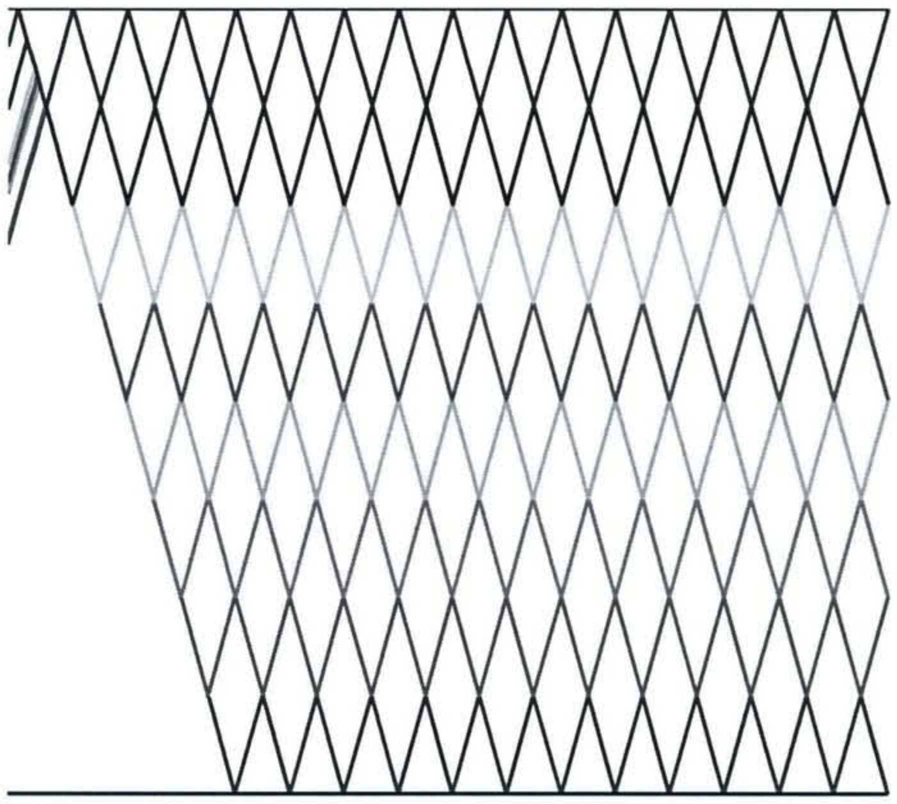


Figure 29: Perimeter surveillance where deadbeat average-consensus occurs.

3.5 Finite-Time Average-Consensus

An average-consensus protocol will invariably require a strongly connected network since each agent must be able to influence the group decision to reflect its initial condition. Obviously, if each agent transmits its initial condition to its neighbors and passes any communication received from others along, then if a strongly connected network is available, eventually all agents will have the complete set of initial conditions from which the average can be computed (clearly, this is not novel; Lynch [57] classifies such an algorithm as trivial). Using this method, all agents will have the information necessary to be in average-consensus after d steps where d is the diameter of the graph. Indeed, it may seem that the restriction to a strongly connected network eliminates any need for an asymptotic protocol. This section will investigate the trade-offs between an asymptotic protocol (such as (22)) and a simple message passing protocol.

The main advantage of an asymptotic consensus protocol is the small amount of bandwidth required - each agent needs only to send its current value. Additionally, there is no need to identify individual agents or know the number of agents in the team. On the other hand, a message passing protocol could keep track of which initial conditions it has sent to each of its neighbors and effectively limit its bandwidth to be the same as the asymptotic protocol, relying instead on repeated interaction to transmit all initial conditions. At each instant, a node's value would be the sample average, i.e. the average of all initial conditions received so far. For large networks, however, the amount of overhead and the complexity may be prohibitive. Perhaps the main advantage of the message passing protocol is the ability to utilize any type of data (not simply continuous real numbers) in any functional way, i.e. agents are not limited to average-consensus but they can come to agreement on any function of the initial conditions.

The message passing protocol effectively emulates a fully connected graph at $k = 0$ by transmitting the required information incrementally. The deadbeat nature of the protocol makes it attractive, especially when speed of convergence is an issue.

An asymptotic protocol will be useful in very large networks and in situations when the value at each node is driven by an external source (agent values are dynamic rather than static) such as distributed Kalman filtering [53]. In many cases, however, a simple message passing scheme may be more attractive due to its deadbeat nature and its ability to handle any data type.

In an effort to quantify the performance of average-consensus as compared to a message passing protocol, we performed a number of Monte-Carlo simulations. We varied team size and available bandwidth to determine under which circumstances and how quickly the average consensus and message passing algorithms would converge. For different bandwidth sizes, the message passing protocol picked a random set of the bandwidth size and passed those values to its neighbors at every time step. For example, in the bandwidth size 2 case, the message passing protocol would pick 2 random values from its vector of known team values and pass those to its neighbors. In the N bandwidth case, an agent would send all values that it has seen to its neighbors. In every bandwidth case, the average-consensus protocol simply took the average of its immediate neighbors and passed only one message to each of its neighbors.

Each simulation consisted of 2000 iterations with a fixed team size and bandwidth. Table 3 shows the results of the tests where each element of the table corresponds to a specific team size/bandwidth test. Each cell contains the mean and standard deviation for the number of iterations required to converge to within a small amount of the final consensus value. The top numbers in a cell correspond to the message passing protocol and the colored numbers correspond to the average consensus algorithm. As can be seen the message passing protocol outperforms the average consensus when there is unlimited bandwidth (i.e. bandwidth size N). In that case, every agent communicates to its neighbors all the initial conditions of the team that it has seen previously. Even for a large team size of 64 agents, the message passing protocol converges an order of magnitude faster than the average-consensus algorithm. As the available bandwidth goes down, however, the average-consensus algorithm shines. Even with a small team size of 16 agents, at bandwidth of 1 message the average-consensus algorithm converges twice as fast as the message passing protocol. We conclude that for equivalent bandwidths much less than the size of the team, the average-consensus algorithm will on average converge faster than a message passing protocol. When there is an abundance of available bandwidth on the order of the team size, then the message passing protocol will be the best choice.

Table 3: Average Iterations to Consensus

		Bandwidth				
		1	2	4	$\frac{N}{2}$	N
Team Size	4	31.69 (17.95)	15.83 (8.348)	9.078 (4.011)	15.98 (8.371)	8.989 (3.92)
		109.9 (17.62)	109.7 (17.83)	109.7 (18.38)	111.2 (18.2)	110.3 (18.24)
	8	190.2 (91.81)	94.1 (45.86)	47.58 (23.05)	46.58 (22.1)	27.95 (12.23)
		333.4 (44.37)	333.5 (44.77)	333.6 (44.51)	331.4 (43.28)	334.3 (46.01)
	16	1052 (350.7)	542.9 (234.9)	272.3 (120.8)	134.4 (59.23)	75.5 (30.2)
		837.4 (115.3)	837.9 (114)	841.8 (115.1)	837.9 (116.5)	839.4 (114.3)
	32	3166 (148.6)	2545 (625.9)	1393 (556.8)	340.9 (145.4)	186.6 (67.65)
		1862 (269.9)	1857 (263.4)	1853 (260.4)	1857 (268.7)	1856 (264.2)
	64	6400 (0)	6388 (95.83)	5447 (1042)	721.2 (283.3)	410.6 (131.4)
		3796 (517.8)	3824 (526.9)	3807 (526.1)	3811 (506.2)	3774 (515.3)

In practical scenarios with wireless radio modems, it will often be the case that the overhead required to begin communication with an agent encourages larger communication packets (due to the cost of switching between neighbors). If the team is known to be moderate sized, the savings in convergence time may be substantial by using a message passing protocol. Finally, we reiterate that consensus algorithms can only come into agreement on continuous valued variables; a message passing scheme allows any function of the initial conditions to be used, including discrete functions such as voting protocols. For very limited communication bandwidth or very large teams, consensus algorithms are a good choice.

3.6 Conclusions

This section has extended the average-consensus results of [35] to allow for networks to switch between instantaneously balanced networks that are strongly connected over every interval T. The discrete-time case has been dealt with explicitly and two asymptotic protocols presented that achieve average-consensus under switching topologies. The notion of deadbeat consensus was investigated with conclusion that general consensus problems may best be solved using a message passing mechanism rather than defining dynamics of the information variable if a strongly connected network can be assumed.

4 Experiments in Cooperative Timing for Miniature Air Vehicles

The effectiveness of unmanned aerial vehicles (UAVs) in recent years has increased dramatically. This increase is due to advances in microcontroller, battery, and sensor technologies. As a result, the use of UAVs, primarily in military applications, has also increased. Larger UAVs, such as Global Hawk, Predator, and Hunter, are currently being used for border patrol, surveillance, and even target prosecution. Smaller microcontollers and sensor packages have also improved the capabilities of miniature aerial vehicles[1], and significant resources have been allocated toward the further development of these technologies [58]. MAVs have a number of applications for which they are especially well suited because of their size and cost. Some of these applications include reconnaissance, battle damage assessment, and surveillance. Although there are many uses for MAVs in commercial or civilian applications, their reliability and capabilities will have to be improved in order to fully transition into the commercial sector.

In recent years, cooperative control for teams of UAVs has been a heavily researched area. Chandler, et al. have explored strategies for dealing with the complexity and coupling inherent in cooperative autonomous UAV teams [59]. They show, by computer simulation, that using network flow and auction algorithms for task assignment yields improved team performance for search missions with varying numbers of team members and tasks to be accomplished.

Much work has also been devoted to path planning for multi-agent teams in high-risk environments. Flint, et al. have developed a model for task assignment and path planning involving both a priori and dynamic information about threats and targets [60]. Simulation results in a three-dimensional environment illustrate the effectiveness of this method. Maza and Ollero present a method for cooperatively searching an area with a heterogeneous team of UAVs taking advantage of the capabilities of the individual agents in the team [61]. The area to be searched is divided into a set of polygons and the assignments for the searching of these polygons is based on which of the UAVs is best suited for the task. Vincent and Rubin have proposed to use swarming methods for cooperatively searching an area [62]. The UAVs fly in a formation that utilizes the sensing capabilities of each team member and easily adapts if any of the members is lost during the mission.

This section extends work presented previously in the area of cooperative timing missions utilizing coordination variables and functions to make path planning and timing decisions [63, 32]. Coordinated timing over a known target can be advantageous in a number of scenarios. One example is the case where a persistent image of a target from a consistent perspective is desired. If the camera footprint for each MAV in a team is known, arrival intervals for the team can be determined such that a persistent image of the target can be obtained. Another scenario is surveillance of a target in a risky environment. A simultaneous arrival of a team of MAVs over the target could be coordinated, thus maximizing the possibility for the team, or at least part of the team, to reach the target. An optimization algorithm based on minimizing fuel consumption was executed to determine paths and velocities for each of the MAVs.

Section 4.1 outlines these two scenarios and describes the issues involved in achieving cooperation for a team of MAVs. Section 4.2 discusses the technical approach taken to address these issues, and Section 4.3 describes of the experimental setup used. The experimental results in Section 4.4 are the primary contribution of this section and illustrate the feasibility of real-time cooperative timing missions for MAVs.

4.1 Problem Overview

The focus of this section is experimental demonstration of cooperative timing scenarios for MAVs, thereby providing practical validation of our cooperative control approach. Two specific missions are considered: simultaneous arrival and cooperative fly-by.

For the simultaneous arrival mission, the objective is for the MAVs composing the team to arrive at the target at the same time from different approach directions. The ability to have multiple vehicles arriving at destination simultaneously is useful for a variety of tactical purposes including formation join-up and target

[1]We consider miniature aerial vehicles to be those with wingspans in the 0.3 m to 2 m range and micro aerial vehicles to be those with wingspans under 0.3 m. Here the abbreviation MAV denotes *miniature* aerial vehicle.

attack. In an attacking situation, simultaneous arrival maximizes the element of surprise and increases the intensity of the attack.

For the cooperative fly-by mission, the objective is for the MAVs to approach the target along the same approach path at specified timing intervals. The ability to have multiple vehicles fly over a target at specified intervals is useful for persistent surveillance or for prosecuting targets where a sequence of tasks is required (e.g., identification, attack, battle damage assessment). We consider the scenario where the heading through the target is predetermined to be along the imaging path so that a persistent image of the target from the same viewing angle can be obtained.

Both missions have several common elements in their planning and execution. During the initial phase of the mission, the MAVs are occupied with a secondary task (i.e. monitoring secondary targets, positioning themselves for the cooperative mission, etc.). Upon command from the operator at the ground station, paths are planned to satisfy the timing constraints for the mission and to minimize the cooperation objective (e.g., minimize fuel cost). Paths are planned based on current MAV and target locations and the desired headings through the target. Upon completion of the pass over the target, the MAVs return to the secondary tasks they were occupied with during the initial phase of the mission. Challenges associated with these missions include real-time cooperative path planning, reliable and timely communication of cooperation information, and accurate path following with significant wind disturbances.

The cooperative timing solutions for the two missions that we will present were obtained from a cooperative timing algorithm that employed coordination variables and coordination functions to represent critical timing information for the MAVs. The experimental results that will be presented involve a team of three MAVs although the approach applies generally to two or more vehicles. This cooperative timing strategy is outlined in the following section together with the path planning and path following approaches that were used.

4.2 Technical Approach

Cooperative Timing Strategy

We employ a cooperative timing strategy that is based on the utilization of coordination variables and coordination functions. The underlying concept of coordination variables and coordination functions is that the minimal information essential to achieving a cooperation objective should be identified and communicated among agents on the team. Minimizing the amount of information being sent improves communication speed and reliability. It also tends to simplify cooperative control calculations by reducing the problem to its essential elements.

Let $\mathbf{x}_i$ define the situation state for the i^{th} vehicle on a team. For the cooperation problems considered in this section, the situation state includes information about the current MAV position, the target position, the desired heading through the target, and the time interval between MAV arrivals (zero for the simultaneous arrival scenario). Additional information about the environment (e.g., wind speed and direction) can also be included if it is available. For a given situation $\mathbf{x}_i$, the set of feasible decisions for the i^{th} vehicle is represented by $\mathcal{U}_i(\mathbf{x}_i)$, and $\mathbf{u}_i \in \mathcal{U}_i$ is the decision variable for the i^{th} vehicle. The choice of the decision variable by each vehicle on the team affects both the existence and the value of the cooperation objective. In the cooperative timing scenarios considered, the decision variable consists of a waypoint path and an average velocity.

The process of cooperation among agents can be viewed as having objectives and constraints. For the simultaneous arrival scenario, the cooperation constraint requires the MAVs to arrive over the target at the same time. For the fly-by scenario, the cooperation constraint requires the MAVs to fly over the target with the second vehicle at a specified time interval behind the first, and the third vehicle arriving at a specified time interval after the second. Cooperation is said to occur if the cooperation constraints are met. In the these scenarios, the cooperation objective is to minimize the battery energy required to complete the mission. The contribution of a vehicle to the team cooperation objective is represented by an influence function, $\phi_i = J_i(\mathbf{u}_i)$. In this case the energy consumption for the i^{th} vehicle (represented by ϕ_i) is a function of the MAV velocity and the waypoint path (represented by $\mathbf{u}_i$).

The coordination variable θ represents the minimal amount information necessary to achieve cooperation. For the simultaneous arrival problem, the coordination variable is the arrival time over the target. For the fly-by problem, the coordination variable is the arrival time of the first MAV and the arrival order for the vehicles. If every vehicle has knowledge of the value of the coordination variable and responds accordingly, cooperative behavior will be achieved by the team. For a vehicle, the value of the coordination variable is related to the decision variable, $\theta_i = f_i(\mathbf{u}_i)$, and defines what the vehicle can do to ensure that cooperation constraints are met. In the scenarios considered in this section, the choice of path and velocity determines MAV energy consumption and arrival time and therefore influences both the cooperation objective and the cooperation constraint. Under the assumptions that battery power is proportional to aerodynamic drag and that the MAV flies at constant speed, the battery energy consumption is given by

$$J_i = c_b v_i L_i$$

where $c_b > 0$ is a constant, v_i is the MAV velocity, and L_i is the length of the waypoint path.

For the cooperative fly-by scenario with three MAVs, the cooperation constraints can be written as

$$\begin{aligned} T_1 &= T_s, \\ T_2 &= T_s + \Delta_2, \\ T_3 &= T_s + \Delta_3, \end{aligned} \tag{24}$$

where T_i is the arrival time of the i^{th} vehicle, Δ_i represents the interval between the arrival of the first and i^{th} vehicles, and $\theta = T_s$ is the coordination variable. Note that for the simultaneous arrival scenario $\Delta_2 = \Delta_3 = 0$.

Critical to the implementation of this approach is the definition of the coordination function. The coordination function models a vehicle's influence on the cooperation objective in terms of what the agent can do to meet the cooperation constraints. The coordination function, $\phi_i(\theta_i)$ is derived from the influence function and coordination variable as

$$\begin{aligned} \phi_i &= J_i(\mathbf{u}_i) \\ &= J_i[f_i^{\dagger}(\theta_i)] \\ &= \phi_i(\theta_i), \end{aligned}$$

where $f_i^{\dagger}$ is the pseudoinverse of f_i. Typically, f_i is not a one-to-one mapping in that numerous values of the decision variable (e.g., waypoint paths, velocities) can result in a single value of the coordination variable (e.g., time over target). The pseudoinverse of f_i is found by taking the value of decision variable that minimizes the influence function J_i, thus creating a one-to-one map between θ_i and $\mathbf{u}$. In other words, when multiple options exist to get the same result, the lowest-cost option is chosen.

Using coordination variables and coordination functions, the battery energy required to complete the mission can be minimized by solving the optimization problem

$$\theta^* = \arg\min_{\theta} \left[\phi_1(\theta) + \phi_2(\theta + \Delta_2) + \phi_3(\theta + \Delta_3)\right],$$

which implies that the constraint in Equation (24) is satisfied. Once a team optimal value for the coordination variable is found, vehicle decisions can be found from the relationship

$$\mathbf{u}_i = f_i^{\dagger}(\theta^*).$$

Figure 30 illustrates the coordination functions calculated for a cooperative timing mission based on input information provided to the algorithm. Because the coordination function for each UAV is monotonically decreasing, the team-optimal arrival time for one of the vehicles will always lie at the right extreme of its coordination function. For the simultaneous arrival scenario, the solution is simply determined by finding the largest arrival time that all three MAVs have in common. For the fly-by scenario, determining the minimum-energy arrival time and order for the team involves a simple search through the right extreme (minimum) values of each coordination function with an evaluation of the team objective to determine the best values.[32]

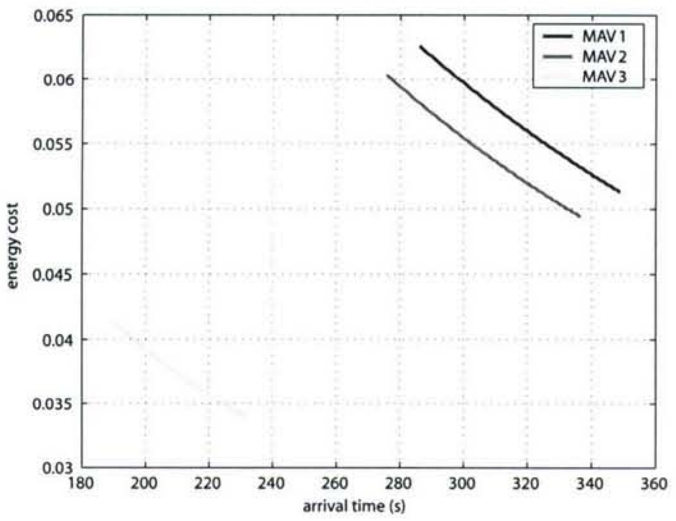


Figure 30: Cooperative timing coordination functions.

Path Planning

In order to simplify the path planning for this initial implementation, some assumptions about the environment were made. The first assumption was that there were no threats to be avoided. The next assumption was that both the target location and the desired fly-by heading were known, which eliminated any initial target observation and detection steps prior to the cooperative timing mission. Because there were no threats to avoid, a straightforward heuristic approach for planning straight-line waypoint paths was utilized. Straight-line paths were easy to plan and simplified both the distance calculations and timing optimization.

Figure 31 shows an example of the waypoint paths used in the experiments. Waypoint 1 for each agent was calculated to be 100 m along the same heading that the agent had when the coordination algorithm was executed. Waypoint 2 was set to be a fixed distance from the known target location along the desired approach path. For the experiments, this distance was between 200 m and 300 m. Allowing this much distance ensured that the transient path following errors associated with transitioning from one straight line path to another would be damped out and that the MAVs would be flying along the desired approach heading when they passed over the target. Waypoint 3 was the target location. Waypoint 4 was collinear with waypoints 2 and 3 on the opposite side of the target from waypoint 2. Waypoints 2, 3, and 4 were the same for each agent in the fly-by scenario and only waypoint 3 was the same in the simultaneous arrival scenario. These three points are referred to as fly-through points. Waypoint 5 was a waypoint to help the agents distance themselves from the target before returning to their final loiter positions, waypoint 6.

In some cases, depending on the initial MAV positions and headings, the coordination functions do not yield a solution that satisfies the cooperative timing constraint. For example, a cooperative fly-by with a spacing interval of 10 s between vehicles could not be accomplished with the coordination functions of Figure 30 due to the time gap between the coordination function of UAV 1 and those of UAVs 2 and 3. In these instances, it became necessary to develop some simple approaches for adding length to one or more paths so that the timing constraints could be met.

Based on the coordination functions, the length to be added to a path was calculated to ensure constraint feasibility. If the length that needed to be added was less than 150 m, the location of the first waypoint was adjusted using the law of cosines so that the total distance would be the original length plus the necessary additional length. If the additional length was longer than 150 m, two additional waypoints were added to the path immediately following the first waypoint. The second waypoint was calculated so that its distance from

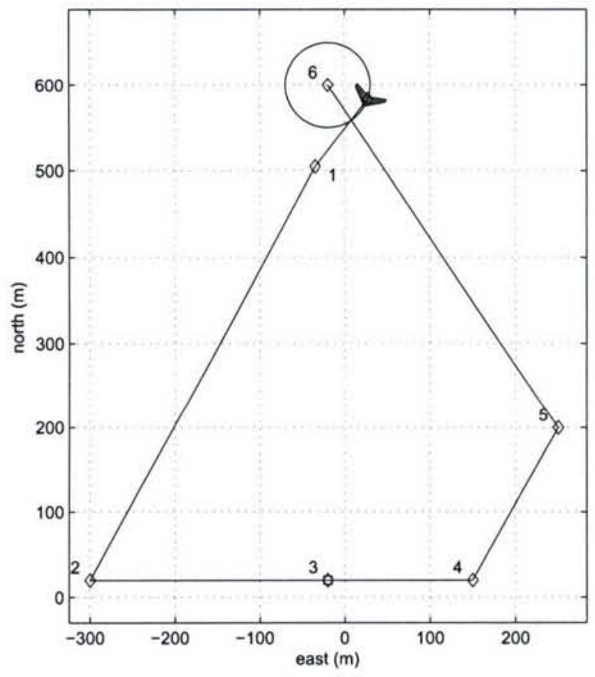


Figure 31: Waypoint path example.

the first waypoint was half of the required additional distance, and the waypoint was placed in a direction directly opposite the target. The third waypoint was placed at the same location as the first, thus effectively adding the required length to the path. The same path shown in Figure 31 with these added points is shown in Figure 32.

Figure 33 shows how the coordination function for UAV 3 was modified by adding length to the original path. With the added length, the coordination function for UAV 3 shifted up and to the right allowing the 10 s interval timing constraint to be satisfied.

Path Following and Velocity Control

For timing missions, preserving path length is critical in order to be able to satisfy the timing constraints. Flying extra distance due to poor path following adds unwanted time to the mission. To address this issue, a vector field path following method has been implemented to enable the MAVs to accurately follow the planned waypoint paths. It has been shown in References [64] and [65] that creating vector fields of desired heading in order to direct the MAV onto the path will result in asymptotically decaying tracking error provided ground track heading and ground speed are used in the control law. Use of this method decreased path following error, and consequently, overall path length error.

Since wind speed is typically 10 to 50 percent of the airspeed for a MAV, wind can have adverse effects on path following and timing constraints. Typically, throttle is controlled based on a desired airspeed. This creates problems when trying to fly timing missions in the presence of wind. To address this issue, the throttle was controlled based on ground speed with a check to make sure that the airspeed did not drop below the MAV stall speed. Throttle control based on ground speed also helps reduce the influence of path length error on timing.

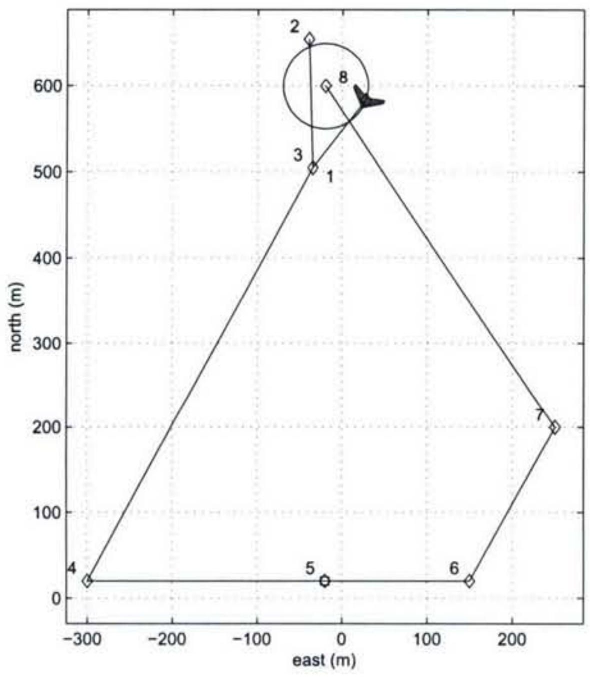


Figure 32: Waypoint path example with added length.

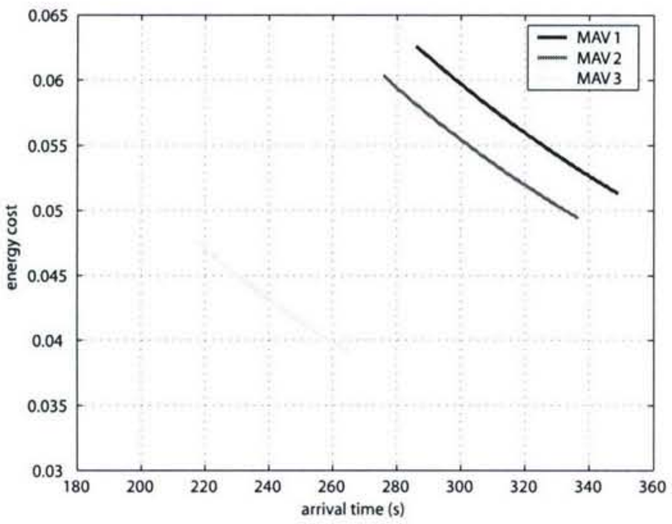


Figure 33: Cooperative timing coordination functions with length added for UAV 1.

4.3 Experimental Setup

Hardware Description

BYU has developed a reliable and robust experimental platform to flight test algorithms for MAVs [17, 66]. Figure 34 shows the key elements of the testbed. The first frame shows BYU's Kestrel autopilot which is equipped with a Rabbit 3400 29 MHz processor, rate gyros, accelerometers, absolute and differential pressure sensors. The autopilot measures 3.8×5.1×1.9 cm and weighs 18 g. The second frame in Figure 34 shows the airframes used for the flight tests reported in this section. The airframe is a 1.1 m wingspan Unicorn EPP foam flying wing, which was selected for its durability, ease of component installation, and flight characteristics. Embedded in the airframe are the Kestrel autopilot, batteries, a 1000 mW, 900 MHz radio modem, a GPS receiver, a video transmitter, and a small analog camera. The third frame in Figure 34 shows the ground station components. A laptop runs the Virtual Cockpit software that interfaces through a communication box to the MAVs. An RC transmitter is used as a stand-by fail-safe mechanism to facilitate safe operations.

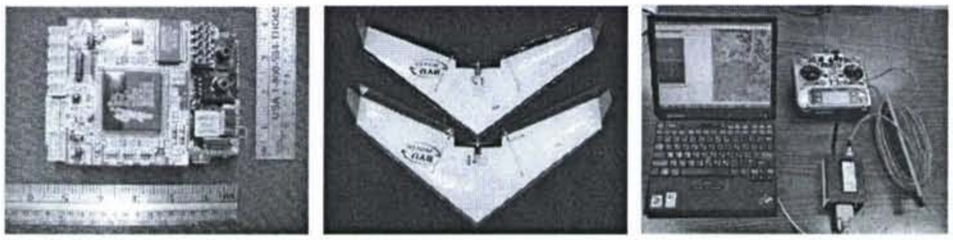


Figure 34: (a) Kestrel autopilot, (b) Unicorn airframes, (c) ground station components.

Cooperative Control Algorithm

The cooperative timing algorithm was implemented as part of the Virtual Cockpit ground station software. The current implementation is centralized, in that the ground station acquired all the information required to do the cooperative timing calculations from the vehicles, performed the calculations, and sent only the paths to be flown and desired arrival times back up to the MAVs. Execution of the algorithm took approximately 1 s, with a significant portion of the time being consumed by communication between the MAVs and the ground station. Once the MAVs received the paths and arrival times, the autopilot controlled the heading using a vector field method [64] and the throttle based on ground speed so that cooperative timing could be achieved. The algorithm used in this section is shown in Table 4.

Table 4: Cooperative timing algorithm.

1	Acquire GPS locations and headings for each MAV
2	Calculate waypoint paths for each MAV
3	Compute range of arrival times for each MAV
4	Compute coordination function for each MAV
5	Based on coordination functions, choose team optimal coordination variable (arrival time of first MAV and arrival order for fly-by and arrival time for simultaneous arrival)
6	Calculate desired arrival time for each MAV
7	Send waypoint paths and desired arrival times to each MAV
8	MAVs fly waypoint paths controlling velocities to achieve cooperation

4.4 Results and Discussion

Simultaneous Arrival

Consecutive simultaneous arrival runs were flown during a 25-minute flight test. Telemetry data from one of the runs is shown in Figure 35. Wind was between 30 and 60 percent of the MAV airspeed during each of the four runs. The average time between the arrival of the first MAV at the target and the arrival of the final MAV was 1.6 s. Range-to-target data is displayed in Figure 36. This illustrates the near simultaneous arrival of the MAVs at the target enabled by the cooperative timing algorithm. Video footage of a simultaneous arrival mission can be viewed by clicking here.

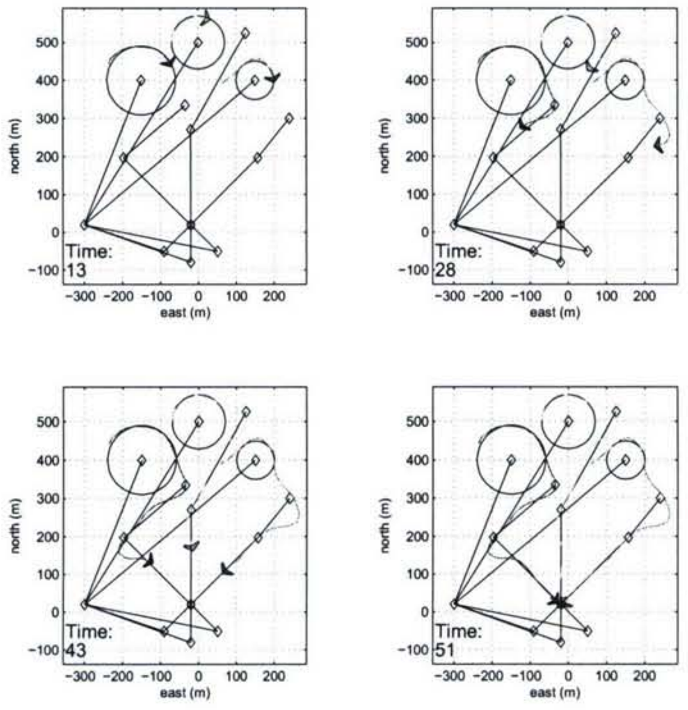


Figure 35: Simultaneous arrival telemetry.

Cooperative Fly-by

The fly by scenario was flown six times during a half-hour flight involving three MAVs. The wind speeds during the flight were between 30 and 60 percent of the MAV airspeed and were from the southwest. The position telemetry from one of the fly-bys is plotted in Figure 37. The desired arrival intervals for this run were 3 s for both intervals and the actual arrival intervals were 3 and 4 s, respectively. The average error in arrival time for the six runs, in spite of the high wind conditions, was approximately 0.6 s. The range to the target versus time for each of the MAVs is shown in Figure 38. The rise in the range curves results from the overshoot that occurs after the first of the fly-through waypoints. This illustrates the arrival of the MAVs at approximately 3 s intervals. Video footage of a cooperative fly-by mission can be viewed by clicking here.

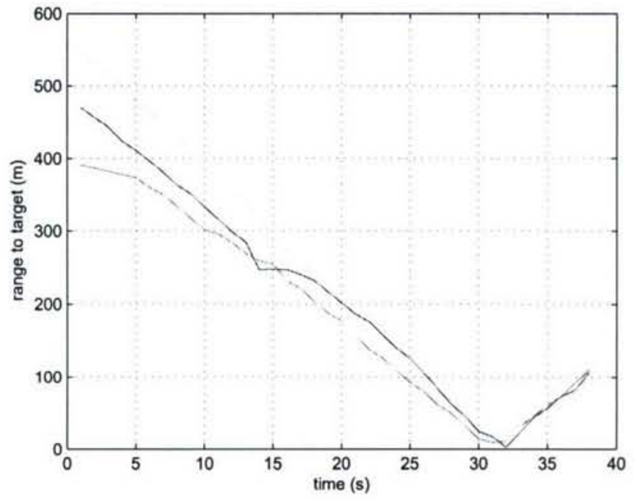


Figure 36: Simultaneous arrival range to target.

Figure 39 illustrates a scenario where the length of the path of one of the MAVs had to be lengthened to satisfy the timing constraint. The MAV that began in the bottom of the frame was considerably closer to the target when the fly-by command was issued and its path length had to be increased for cooperation to

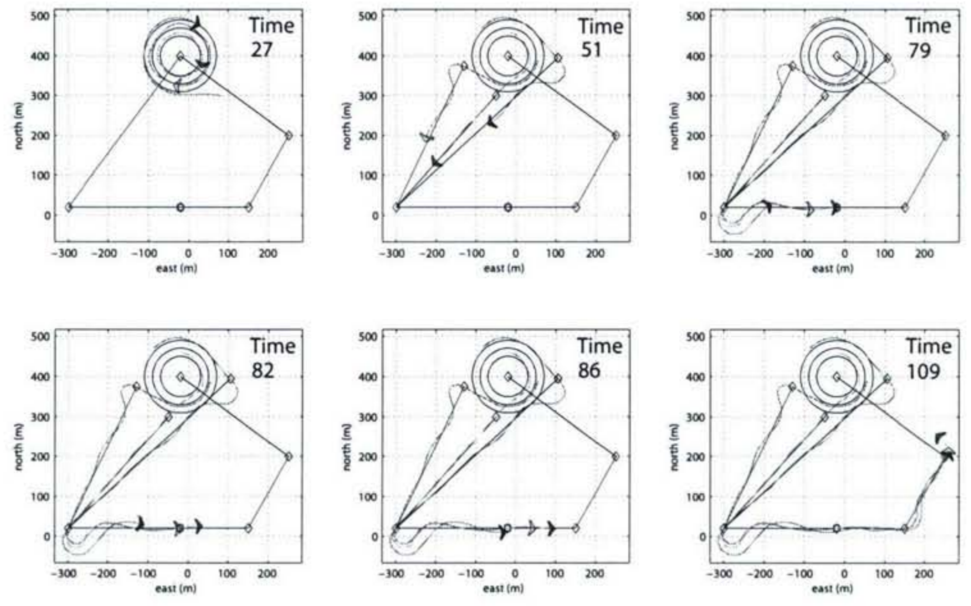


Figure 37: Fly-by telemetry, 3 s arrival intervals.

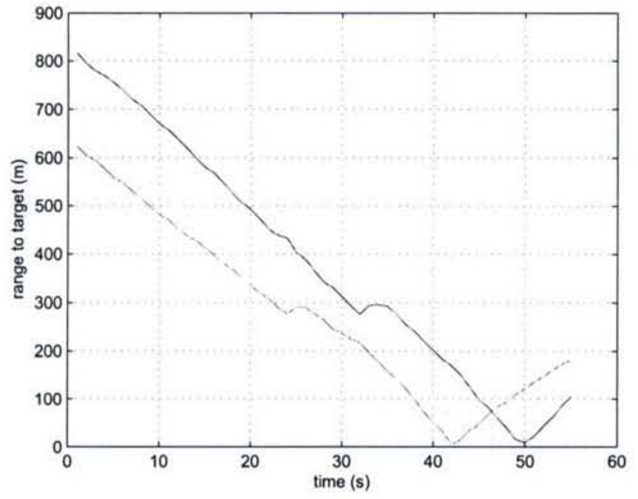


Figure 38: Fly-by range to target, 3 s arrival intervals.

occur. Two waypoints were added after the first waypoint using the method described in Section 4.2. The first two MAVs arrived at the desired arrival times while the third MAV was delayed slightly due to the strong head wind it faced during most of the run.

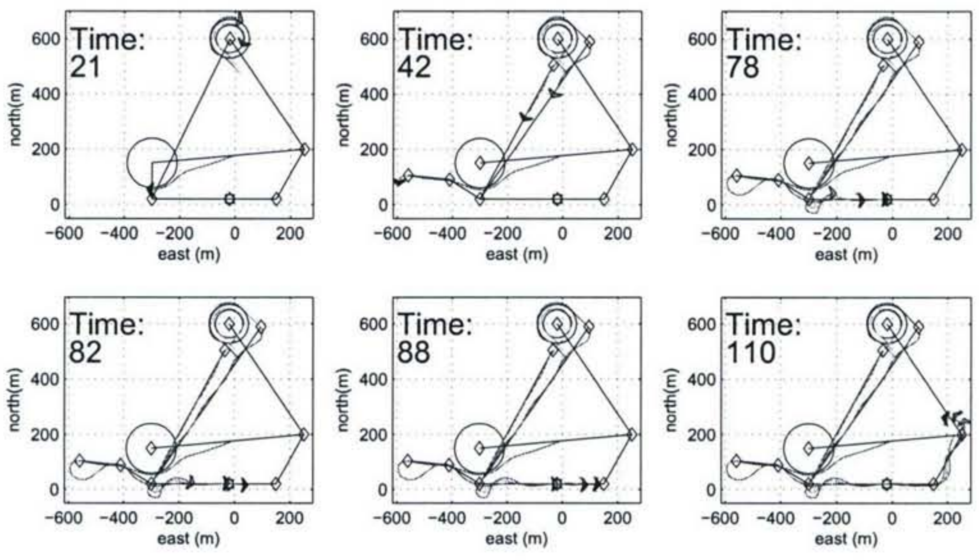


Figure 39: Fly-by telemetry with path lengthened for MAV 1.

Although the timing errors for both simultaneous arrival and cooperative fly-by are relatively small, there are a few possible explanations for the errors that occurred. One of the more obvious reasons was the presence of the high relative wind speeds. If the desired groundspeed was in the upper end of the feasible airspeed envelope (20 m/s) while flying into a headwind, it was difficult for timing constraints to be met. Another issue was synchronizing the timing on each of the autopilots. Each autopilot had its own timer that started when the list of waypoints finished uploading from the ground. This took about 1 s for each of the three agents and if it took longer for one MAV to finish downloading the information than another, the start times were different. Transient behavior involved in matching the desired groundspeed as it changed and GPS error could also have influenced proper timing.

4.5 Conclusion

Cooperative timing missions for miniature aerial vehicles have been explored in this section. The experimental results show the feasibility of real-time coordination for teams of MAVs. Three MAVs were flown in two different flights and multiple cooperative timing missions were run during each flight. Both coordinated fly-by scenarios and simultaneous arrival scenarios were run where the coordination occurred on the ground station using an algorithm based on coordination functions and coordination variables. These results illustrate the repeatability and reliability of this algorithm even in the presence of high winds relative to MAV airspeed.

References

[1] J. Clark and R. Fierro, "Cooperative hybrid control of robotic sensors for perimeter detection and tracking," in *Proceedings of the American Control Conference*, 2005.

[2] B. A. White, A. Tsourdos, I. Ashokoraj, S. Subchan, and R. Zbikowski, "Contaminant cloud boundary monitoring using UAV sensor swarms," *AIAA Journal of Guidance, Control, and Dynamics*, (submitted).

[3] A. L. Bertozzi, M. Kemp, and D. Marthaler, "Determining environmental boundaries: Asynchronous communication and physical scales," in *Proceedings of the Block Island Workshop on Cooperative Control*, Springer-Verlag Series: Lecture Notes in Control and Information Sciences, 2004.

[4] D. W. Casbeer, S.-M. Li, R. W. Beard, T. W. McLain, and R. K. Mehra, "Forest fire monitoring using multiple small UAVs," in *Proceedings of the American Control Conference*, 2005.

[5] D. W. Casbeer, D. B. Kingston, R. W. Beard, T. W. McLain, S.-M. Li, and R. Mehra, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of System Sciences*, vol. 36, pp. 351–360, May 2006.

[6] R. T. Laird, H. R. Everett, G. A. Gilbreath, T. A. Heath-Pastore, and R. S. Inderieden, "MDARS multiple robot host architecture," in *Association of Unmanned Vehicle Systems, 22nd Annual Technical Symposium and Exhibition*, 1995.

[7] H. R. Everett, "Robotic security systems," *IEEE Instrumentation & Measurement Magazine*, vol. 6, pp. 30–34, Dec. 2003.

[8] S. Young, M. Forshaw, and M. Hodgetts, "Image comparison methods for perimeter surveillance," in *Proceedings of the International Conference on Image Processing and Its Applications*, 1999.

[9] J. O. Peralta and M. T. C. de Peralta, "Security PIDS with physical sensors, real-time pattern recognition, and continuous patrol," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 32, pp. 340–346, Nov. 2002.

[10] A. S. Barry and J. Czechanski, "Ground surveillance radar for perimeter intrusion detection," in *Proceedings of the Digital Avionics Systems Conference*, 2000.

[11] D. Marthaler and A. L. Bertozzi, "Tracking environmental level sets with autonomous vehicles," in *Recent Developments in Cooperative Control and Optimization*, Kluwer Academic Publishers, 2004.

[12] Space and Naval Warfare Systems Command, "Mobile detection assessment and response system (MDARS)."

[13] M. Kemp, A. L. Bertozzi, and D. Marthaler, "Multi-UUV perimeter surveillance," in *Proceedings of the IEEE/OES Autonomous Underwater Vehicles Conference*, 2004.

[14] C. H. Hsieh, Z. Jin, D. Marthaler, B. Q. Nguyen, D. J. Tung, A. L. Bertozzi, and R. M. Murray, "Experimental validation of an algorithm for cooperative boundary tracking," in *Proceedings of the American Control Conference*, 2005.

[15] S. Susca, S. Martinez, and F. Bullo, "Monitoring environmental boundaries with a robotic sensor network," *IEEE Transactions on Control Systems Technology*, (accepted for publication).

[16] W. Ren, R. W. Beard, and T. W. McLain, *Cooperative Control*, vol. 309, ch. Coordination Variables and Consensus Building in Multiple Vehicle Systems, pp. 171–188. Springer-Verlag Series: Lecture Notes in Control and Information Sciences, 2004.

[17] R. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. Goodrich, "Autonomous vehicle technologies for small fixed wing UAVs," *AIAA Journal of Aerospace Computing, Information, and Communication*, vol. 2, pp. 92–108, Jan. 2005.

[18] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative timing missions," in *Proceedings of the American Control Conference*, 2003.

[19] W. Ren and R. W. Beard, "Consensus of information under dynamically changing interaction topologies," in *Proceedings of the American Control Conference*, 2004.

[20] D. B. Kingston and R. W. Beard, "Discrete-time average-consensus under switching network topologies," in *Proceedings of the American Control Conference*, 2006.

[21] D. B. Kingston, "Implementation issues of real-time trajectory generation," Master's thesis, Brigham Young University, 2004.

[22] A. Robertson, G. Inalhan, and J. P. How, "Formation control strategies for a separated spacecraft interferometer," in *Proceedings of the American Control Conference*, June 1999.

[23] R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A coordination architecture for formation control," *IEEE Transactions on Control Systems Technology*, vol. 9, pp. 777–790, Nov. 2001.

[24] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating UAVs," in *Cooperative Control: Models, Applications and Algorithms*, pp. 1–19, Conference on Coordination, Control and Optimization, Nov. 2001.

[25] R. Emery, K. Sikorski, and T. Balch, "Protocols for collaboration, coordination and dynamic role assignment in a robot team," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3008–3015, May 2002.

[26] G. Inalhan, D. M. Stipanovic, and C. J. Tomlin, "Decentralized optimization with application to multiple aircraft coordination," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 1147 – 1155, 2002.

[27] W. Kang, N. Xi, and A. Sparks, "Formation control of autonomous agents in 3D workspace," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1755–1760, Apr. 2000.

[28] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 2968–2973, Dec. 2001.

[29] P. Ogren, M. Egerstedt, and X. Hu, "A control Lyapunov function approach to multiagent coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 847–851, Oct. 2002.

[30] D. J. Stilwell and B. E. Bishop, "Platoons of underwater vehicles," *IEEE Control Systems Magazine*, vol. 20, pp. 45–52, Dec. 2000.

[31] H. G. Tanner, V. Kumar, and G. J. Pappas, "The effect of feedback and feedforward on formation ISS," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3448–3453, May 2002.

[32] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative timing missions," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 28, pp. 150–161, Jan. 2005.

[33] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, Sept. 2004.

[34] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, pp. 988–1001, June 2003.

[35] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1520–1533, Sept. 2004.

[36] L. Moreau, "Stability of multiagent systems with time-dependent communication links," in *IEEE Transactions on Automatic Control*, 2005.

[37] Z. Lin, M. Broucke, and B. Francis, "Local control strategies for groups of mobile autonomous agents," *IEEE Transactions on Automatic Control*, pp. 622–629, 2004.

[38] W. Ren and R. W. Beard, "Consensus seeking in multi-agent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 5, pp. 655–661, May 2005.

[39] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2004.

[40] P. Chandler, S. Rasumussen, and M. Pachter, "UAV cooperative path planning," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2000.

[41] R. W. Beard, T. W. McLain, M. Goodrich, and E. P. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 911–922, Dec. 2002.

[42] T. McLain and R. Beard, "Cooperative rendezvous of multiple unmanned air vehicles," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2000.

[43] B. Barshan and H. F. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 328–342, June 1995.

[44] M. E. Cannon, "Integrated GPS-INS for high-accuracy road positioning," *Journal of Surveying Engineering*, vol. 118, pp. 103–117, Nov. 1992.

[45] S. Ronnback, "Development of an INS/GPS navigation loop for an UAV," Master's thesis, Luleå University of Technology, 2000.

[46] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, vol. 64 of *Mathematics in Science and Engineering*. New York, New York: Academic Press, Inc., 1970.

[47] F. L. Lewis, *Optimal Estimation: With an Introduction to Stochastic Control Theory*. New York, New York: John Wiley & Sons, 1986.

[48] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer-Verlag New York, Inc., 2001.

[49] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1987.

[50] J. Wolfowitz, "Products of indecomposable, aperiodic, stochastic matrices," *Proceedings of the American Mathematical Society*, vol. 15, pp. 733–736, 1963.

[51] W. J. Rugh, *Linear System Theory*. Englewood Cliffs, New Jersey: Prentice Hall, 2nd ed., 1996.

[52] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, (submitted).

[53] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Approximate distributed Kalman filtering in sensor networks with quantifiable performance," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, 2005.

[54] R. Olfati-Saber, "Distibuted Kalman filter with embedded consensus filters," in *Proceedings of the IEEE Conference on Decision and Control*, 2005.

[55] W. Ren, R. W. Beard, and D. B. Kingston, "Multi-agent Kalman consensus with relative uncertainty," in *Proceedings of the American Control Conference*, 2005.

[56] D. B. Kingston, R. S. Holt, R. W. Beard, T. W. McLain, and D. W. Casbeer, "Decentralized perimeter surveillance using a team of UAVs," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2005.

[57] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996.

[58] "Unmanned Aerial Vehicle Roadmap 2002–2007," tech. rep., Office of the Secretary, U.S. Department of Defense, Dec. 2002.

[59] P. R. Chandler, M. Pachter, D. Swaroop, J. M. Fowler, J. H. Howlett, S. Rasmussen, C. Schumacher, and K. Nygard, "Complexity in UAV cooperative control," in *Proceedings of the American Control Conference*, May 2002.

[60] M. Flint, E. Fernandez-Gaucherand, and M. Polycarpou, "Cooperative control for UAVs searching risky environments for targets," in *Proceedings of the IEEE Conference on Decision and Control*, Dec. 2003.

[61] I. Maza and A. Ollero, "Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms," in *7th International Symposium on Distributed Autonomous Robotic Systems*, June 2004.

[62] P. Vincent and I. Rubin, "A framework and analysis for cooperative search using UAV swarms," in *Proceedings of the 2004 ACM symposium on Applied Computing*, Mar. 2004.

[63] T. W. McLain and R. W. Beard, "Unmanned air vehicle testbed for cooperative control experiments," in *Proceedings of the American Control Conference*, pp. 5327–5331, June 2004.

[64] D. Nelson, T. McLain, and R. Beard, "Vector field path following for small unmanned air vehicles," *IEEE Transactions on Robotics and Automation*, 2005. (submitted).

[65] D. R. Nelson, "Cooperative control of miniature air vehicles," Master's thesis, Brigham Young University, Department of Mechanical Engineering, Dec. 2005.

[66] R. W. Beard, D. Lee, S. Thakoor, and S. Zornetzer, "A new approach to observation of descent and landing of future Mars mission using bioinspired technology innovations," *AIAA Journal of Aerospace Computing, Information, and Communication*, vol. 2, pp. 65–91, Jan. 2005.

Personnel Supported

Tim McLain, Professor, Mechanical Engineering
Randy Beard, Professor, Electrical and Computer Engineering
Derek Kingston, Graduate Student, Electrical and Computer Engineering
Wei Ren, Graduate Student, Electrical and Computer Engineering
Derek Nelson, Graduate Student, Mechanical Engineering
Joshua Redding, Graduate Student, Mechanical Engineering
Steven Hansen, Graduate Student, Mechanical Engineering

Interactions

Over the course of this project, we have had regular interactions with Jennifer Wilds (AFRL/MNAV), Rob Murphey (AFRL/MNGN), Johnny Evers (AFRL/MNGN), Andrew Sparks (AFRL/VACA), Corey Schumacher (AFRL/VACA), Phillip Chandler (AFRL/VACA), and Siva Banda (AFRL/VACA).

Students Joshua Redding, Derek Kingston, Steve Olsen, and Jeff Saunders have spent summers at AFRL/VACA working under the direction of Andrew Sparks, Corey Schumacher, and Phillip Chandler.

Randy Beard recently completed a year-long NRC Fellowship at AFRL/MNGN working with Rob Murphey and Johnny Evers.

Transitions

See attached spreadsheet.

Honors and Awards Received

Tim McLain, Finalist, 2007 Stoel Rives Utah Innovation Awards, UAVs for Sensing, Data and Images, with Flying Sensors, Inc.

Tim McLain, AIAA Nominee, Utah Engineering Educator of the Year, 2007

Randy Beard and Tim McLain received the 2006 BYU Technology Transfer Award.

Tim McLain, Outstanding Faculty Award, 2005, BYU Mechanical Engineering Department.

Randy Beard and Tim McLain, Finalists, 2004 Stoel Rives Utah Innovation Awards, BYU Kestrel Autopilot

Randy Beard, BYU Young Scholar Award, 2004-2007

Tim McLain, BYU Young Scholar Award, 2003-2006

Publications

R. Beard and T. McLain, "Multiple UAV Cooperative Search under Collision Avoidance and Limited Range Communication Constraints," *Proc. of the IEEE Conf. on Decision and Control*, December 2003.

R. Beard and V. Stepanyan, "Synchronization of Information in Distributed Multiple Vehicle Coordinated Control," *Proc. of the IEEE Conf. on Decision and Control*, December 2003.

W. Ren and R. Beard, "CLF-based Tracking Control for UAV Kinematic Models with Saturation Constraints," *Proc. of the IEEE Conf. on Decision and Control*, December 2003.

W. Ren, R. Beard, and T. McLain, "Coordination Variables and Consensus Building in Multiple Vehicle Systems," *Proc. of the Block Island Workshop on Cooperative Control*, Springer-Verlag Series: Lecture Notes in Control and Information Sciences, 2004.

D. Walker, T. McLain, and J. Howlett, "Cooperative UAV Target Assignment Using Distributed Calculation of Target-Task Tours," *Theory and Algorithms for Cooperative Control*, World Scientific, 2004.

T. McLain and R. Beard, "Unmanned Air Vehicle Testbed for Cooperative Control Experiments," *Proc. of the American Control Conf.*, June 2004.

R. Christiansen, D. Johansen, T. McLain, and R. Beard, "Initial Experiments in the Cooperative Control of Unmanned Air Vehicles," *Proc. of the AIAA Unmanned Unlimited Conference*, AIAA paper no. 2004-6533, September 2004.

W. Ren and R. Beard, "Trajectory Tracking for Unmanned Air Vehicles with Velocity and Heading Rate Constraints," *IEEE Trans. on Control Systems Technology*, pp. 706-716, September 2004.

R. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. Goodrich,"Autonomous Vehicle Technologies for Small Fixed-wing UAVs," *Journal of Aerospace Computing, Information, and Communication*, pp. 92-108, January 2005.

T. McLain and R. Beard, "Coordination Variables, Coordination Functions, and Cooperative Timing Missions," *AIAA Journal of Guidance, Control, and Dynamics*, pp. 150-161, January 2005.

W. Ren, R. Beard, "Consensus Seeking in Multi-agent Systems Using Dynamically Changing Interaction Topologies," *IEEE Transactions on Automatic Control,* vol. 50, no. 5, pp. 655-661, May 2005.

W. Ren and R. Beard, "A Survey of Consensus Problems in Multi-agent Coordination," *Proc. of the American Control Conference*, June 2005.

W. Ren, R. Beard, and D. Kingston, "Multi-agent Kalman Consensus with Relative Uncertainty," *Proc. of the American Control Conference*, June 2005.

D. Kingston, W. Ren, and R. Beard, "Consensus Algorithms are Input-to-State Stable," *Proc. of the American Control Conference*, June 2005.

D. Kingston, R. Holt, R. Beard, T. McLain, and D. Casbeer, "Decentralized Perimeter Surveillance Using a Team of UAVs," *Proc. AIAA Guidance, Navigation, and Control Conf.*, AIAA-2005-5831, Aug 2005.

J. Howlett, T. McLain, and Goodrich, M. "Learning Real-Time A* Path Planner for Unmanned Air Vehicle Target Sensing." *AIAA J. Aerospace Computing, Information, and Communication,* vol. 3, no. 3, pp. 108-122, Mar 2006.

D. Casbeer, D. Kingston, R. Beard, T. McLain, S. Li, and R. Mehra, "Forest Fire Surveillance Using a Team of Small Unmanned Air Vehicles," *Int. J. of Systems Science, Special Issue on Cooperative Control Approaches for Multiple Mobile Robots*, vol. 37, no. 6, pp. 351–360, May 2006. Invited submission.

J. Jackson, A. Eldredge, D. Nelson, S. Griffiths, T. McLain, and R. Beard, "Miniature Air Vehicle Cooperative Timing Missions," *Video Proc. IEEE Conf. on Robotics and Automation,* May 2006.

D. Nelson, B. Barber, T. McLain, and R. Beard, "Vector Field Path Following for Small Unmanned Air Vehicles," *Proc. American Control Conf.,* pp. 5788–5794, Jun 2006.

J. Redding, T. McLain, R. Beard, and C. Taylor, "Vision-based Target Localization from a Fixed-wing Miniature Air Vehicle," *Proc. American Control Conf.,* pp. 2862–2867, Jun 2006.

D. Kingston and R. Beard, "Discrete-Time Average-Consensus under Switching Network Topologies," *Proc. American Control Conf.,* pp. 3551–3556, Jun 2006.

R. Beard, T. McLain, D. Nelson, and D. Kingston, "Decentralized Cooperative Aerial Surveillance Using Fixed-Wing Miniature UAVs," *Proc. of the IEEE,* vol. 94, no. 7, pp. 1306-1323, Jul 2006.

D. Nelson, B. Barber, T. McLain, and R. Beard, "Vector Field Path Following for Miniature Air Vehicles," *IEEE Transactions on Robotics,* vol. 23, no. 3, pp. 519-529, Jun 2007.

D. Barber, J. Redding, T. McLain, R. Beard, and C. Taylor, "Vision-based Target Geo-location Using a Fixed-wing Miniature Air Vehicle," *Journal of Intelligent and Robotic Systems,* vol. 47, no. 4, pp. 361-382, Dec 2006.

W. Ren, R. Beard and E. Atkins, "Information Consensus and its Applications in Multi-vehicle Cooperative Control," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 71-82, Apr 2007.

D. Nelson, T. McLain and R. Beard, "Experiments in Cooperative Timing for Miniature Air Vehicles," *AIAA Journal of Aerospace Computing, Information, and Communication*, (to appear).

D. Kingston and R. Beard, "UAV Splay State Configuration for Moving Targets in Wind," *Conference on Cooperative Control and Optimization*, Springer-Verlag Series: Lecture Notes in Control and Information Sciences, (to appear).

W. Ren, R. Beard and D. Kingston, "Kalman Consensus Strategies and Their Application to Cooperative Control," *Journal of Robotic and Intelligent Systems*, (in review).

D. Kingston, R. Holt and R. Beard, "Decentralized Perimeter Surveillance Using a Team of UAVs," *Journal of Guidance, Control, and Dynamics*, (in review).

Acknowledgment/Disclaimer

This work was sponsored (in part) by the Air Force Office of Scientific Research, USAF, under grant/contract number FA9550-04-1-0209. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

PI Name	PI Organization	AFOSR PM Name	Title of Research Effort	Follow-on Use	Customer Name	Contact Person	Results	Description of Results	Application or Potential Application
Beard/McLain	Brigham Young Univ.	Sharon Heise	Real-Time Trajectory Generation for Autonomous Nonlinear Flight Systems	transition	AF BATCAM, AF NightHawk, Army TAC MAV programs	I don't know the contacts. Todd Titensor (see below) can provide	product	800 autopilots (and associated control software) sold to AF and Army to date	Autopilots for small UAVs
Beard/McLain	Brigham Young Univ.	Sharon Heise	Real-Time Trajectory Generation for Autonomous Nonlinear Flight Systems	transfer	Procerus Technologies	Todd Titensor, toddt@procerusuav.com	product	autopilots sold commercially to many universities, research organizations, and defense contractors (Raytheon, Lockheed	Autopilots for small UAVs
McLain/Beard	Brigham Young Univ.	Sharon Heise, Scott Wells	Cooperation and Consensus Seeking for Teams of Unmanned Air Vehicles	transfer	Procerus Technologies	Todd Titensor, toddt@procerusuav.com	product	target localization, target tracking capabilities for small UAVs. Procerus is working hard to transition this capability into	target localization, persistent surveillance
McLain/Beard	Brigham Young Univ.			transfer	Procerus Technologies	Todd Titensor, toddt@procerusuav.com	product	vision-based target prosecution	prosecution of fixed and moving targets using small, fixed-wing UAVs. There is tremendous interest in this capability currently. Although this wasn't funded by the AF directly, I include it because without our funding from the AF on other